译 者 的 话

最优化技术在国民经济的各个领域里获得了广泛的应用。近年来，以遗传算法、模拟退火、禁忌搜索以及人工神经网络为代表的智能化优化技术发展迅速，受到人们的普遍关注。其中，遗传算法以其优良的计算性能和显著的应用效果而特别引人注目。

遗传算法将生物进化的原理与最优化技术和计算机技术结合起来，创造了一种全新的优化方法，开辟了一个全新的研究领域。更为难能可贵的是，遗传算法在工业工程、经济管理、交通运输、工业设计等许多领域里获得了广泛的应用。

美国 John Wiley & Sons 出版社 1996 年出版的日本足利工业大学的玄光男(Mitsuo Gen)和程润伟(Runwei Cheng)的英文著作《遗传算法与工程设计》是一本从计算和应用的角度介绍遗传算法的新书。本书在介绍遗传算法的基本理论和概念的基础上，着重讨论了遗传算法的计算技术。书中用统一的符号和术语介绍了近年来散见于文献中的遗传算法的各种应用问题和计算方法，理论上深入浅出，应用上脚踏实地，堪称为一本理论与实际完美结合的好书。书中的内容包括遗传算法的主要构成及其在工业工程和制造系统设计中的优化问题中的应用。本书可作为高等院校的工业工程、管理科学、运筹学、计算机科学和人工智能专业的本科生和研究生的教科书。对于系统分析员、运行研究员、管理科学家，以及在工业工程和运筹学领域里从事研究和应用的专业技术人员，本书也不失为一本内容丰富且实用性较强的参考书。

为了尽早将此书奉献给我国的广大读者，推进我国遗传算法的研究和应用，在原书作者的支持和帮助下，我们及时将此书译成了中文，并在科学出版社出版。在此谨向原书作者玄光男、程润伟博士，表示衷心的感谢！

译者在国家自然科学基金的资助下正在从事智能优化方法及其在生产计划调度中应用的研究工作，本书的翻译出版是我们研究工作的一部分。在此也向支持我们这项研究的国家自然科学基金委员会和东北大学领导表示诚挚的谢意！

本书共分十章，前言和第一至第三章由汪定伟翻译；第四至第七章由唐加福翻译；第八至第十章由黄敏翻译。最后由汪定伟校阅、定稿。由于我们水平所限，译本中错误和疏漏之处在所难免，恳请广大读者指正。

译者

1998 年 9 月于东北大学，沈阳

前　言

遗传算法(GA)是基于进化论的原理发展起来的一种广为应用的、高效的随机搜索与优化的方法。在过去的几年里,遗传算法学界将其注意力集中在工业工程中的优化问题上,从而形成了遗传算法的研究与应用相结合的新的主体。

本书的内容包括遗传算法的主要构成及其在工业工程和制造系统设计中的难解优化问题中的应用。本书可作为专业人员的自学用书,或作为大学和高等院校的工业工程、管理科学、运筹学、计算机科学和人工智能专业的本科生和研究生的教科书。此外,对于系统分析员、运行研究员、管理科学家及工程师,以及一切面临着工业工程与运筹学中的难解的最优化问题挑战的专业人员,本书则是一本内容丰富且实用性强的参考书。虽然近年来已经出版了几本十分不错的遗传算法的专著,但本书却是对遗传算法在工业工程和运筹学中的应用状况进行统一描述和全面综述的第一本书。

在本书中,我们总结了自 1992 年以来遗传算法及相关研究的成果,并按算法的类别对选材作了重新组织。但是,由于目前人们对遗传算法的了解还远不成熟,因此本书没有对现有的遗传算法的理论展开深入讨论。我们希望读者能从这些用遗传算法求解工业工程和运筹学的实际问题的方法中学到有用而必要的知识和技术,从而获得益处。

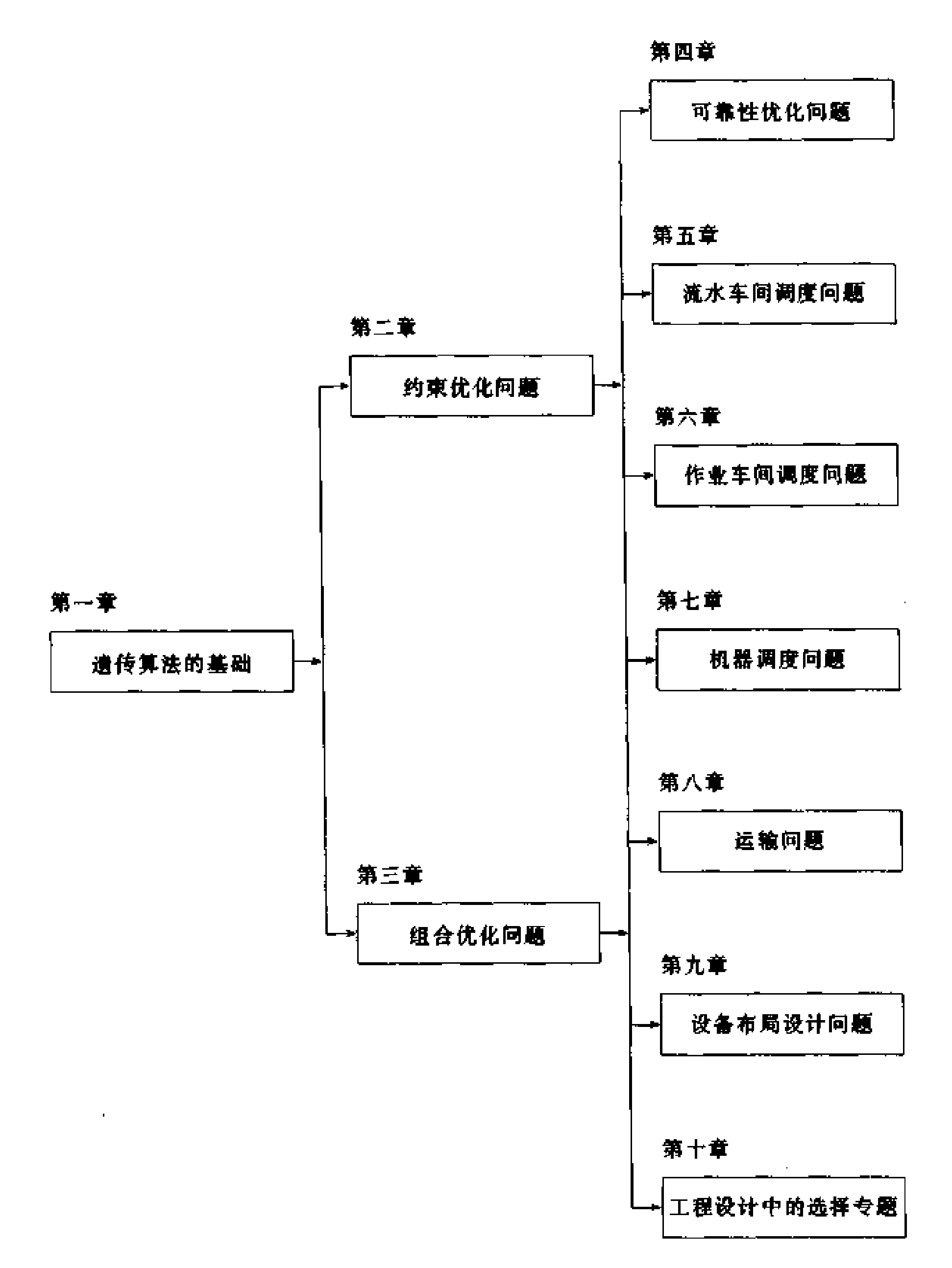
众所周知,对于初学者来说绝大多数遗传算法的专业论文是不易读懂的,因为每个作者都习惯于使用自己的表达方式、术语和符号。本书则统一了所有选材的术语和符号,纠正了其中的错误;特别注重表达的清晰、易懂和可读性。所有的算法都尽可能用自然的语言、图表和计算例子来说明,而尽量避免使用难懂的技术术语。

用本书作为课程教材是十分合适的,因为书中没有采用深奥的数学,绝大多数内容要求的数学未超过大学低年级的水平。对于各个问题,本书主要介绍问题的背景、历史、定义、公式,以及其他求解技术的研究进展。介绍的重点则是应用遗传算法求解问题的新的研究成果。本书自成体系,各专题相关的领域知识也包含在本书之内。

本书选材组织有很大的灵活性。第一至第三章为读者提供了必要的基础知识。余下七章几乎都是独立成章的,但它们都以前三章的材料为基础。全书的体系如下页图所示。

第一章是关于遗传算法的基本概念的。其中介绍的遗传算法的修正的一般结构能更好地反映遗传算法的现行实际情况。对编码问题的深入讨论则为读者对遇到的实际问题设计适当的染色体描述方式提供了一些有助益的观点。此外,本章还从算法的角度对选择策略作了全面的综述。

众所周知,实际生活中许多工业工程的设计问题是非常复杂的,很难用传统的优化方法来解决。近年来,遗传算法作为一种新的优化技术已经受到了广泛的注意。第二章讨论遗传算法在解无约束优化、非线性规划、随机规划、目标规划,以及区间规划等问题中的应用。第三章说明如何解组合优化问题,包括背包问题、二次指派问题、最小支撑树问题、旅行商问题,以及影片递送问题。当然,从这些问题中发展起来的遗传算法的技术对于其他



组合优化问题同样也是适用的。特别是旅行商问题通常作为组合优化难题的典范，许多新方法都用它来测试检验。由于许多难解的工程优化问题都可归结为约束优化或组合优化问题，因此，前三章可以作为读者进一步阅读后续章节中感兴趣问题的遗传算法的基础。

第四章介绍如何解可靠性优化问题，包括具有多种故障模式的冗余系统的可靠性问题。这类问题要考虑冗余单元和替代设计，模糊的目标和约束，以及冗余的混合元件的问题。

第五章讨论如何解流水车间的排序问题。第六章则综合介绍近年来用遗传算法求解作业车间调度问题的文献。作业车间问题在组合优化中是最难的，它吸引了许多遗传算法的研究者的兴趣。这一章里，我们的注意力放在作业车间调度的基因表达方案上。用遗传算法解作业车间问题的研究为约束组合优化问题提供了丰富的经验。所有由作业车间调度发展起来的技术对于现代柔性制造系统的其他调度问题以及其他的组合优化问题或许

都是有用的。第七章讨论如何应用遗传算法求解其他非规则的调度准则下的单机或多机调度问题。机器调度问题是一个充满希望的研究领域，它在制造系统、后勤供应和计算机体系结构等领域里有着广泛的应用。

第八章研究如何解交通问题，包括双判据线性交通问题、双判据整体交通问题，以及模糊多判据整体交通问题。在这一章发展起来的技术可以推广到其他多判据（多目标）优化问题中。

第九章说明遗传算法能够成功地用到各种机器与设备布置的问题中。20 年来，设备布置设计一直是一个多学科关注的课题。由于设备布置问题的组合优化的本质，遗传算法已成为解决实际规模的布置问题的最有希望的方法之一。

第十章讨论几个工业工程设计的选题，包括有资源约束的工程排序问题、定址-分配问题、有障碍的定址-分配问题，以及生产计划问题。这些问题大多比前几章讨论的问题要复杂得多。这些材料能加深读者对遗传算法的理解，提高他们应用遗传算法的能力，并激发他们将遗传算法应用到自己的工作中去。我们希望本书能为遗传算法的普及、提高和更广泛的应用做出贡献。

我们首先衷心地感谢 John Wiley & Sons 出版社“工程设计与自动化丛书”的编辑，*Engineering Design and Automation* 杂志的主编，Louisville 大学的 Hamid R. Parsaei 博士，感谢他给我们提供了为这个新丛书写这本书的机会。我们还衷心地感谢 *Computers & Industrial Engineering* 杂志的主编，Iowa 大学的 Hamed K. Eldin 博士，感谢他为作者之一提供了在上述杂志中主编“遗传算法与工业工程专集”的机会。这本专集激发了我们面向工业工程研究遗传算法的兴趣和干劲。

本书来自我们和我们的同事和朋友的大量讨论。为此我们衷心感谢：Kansas 州立大学的 C. L. Hwang 博士和 Frank A. Tillman 博士，Pennsylvania 州立大学的 Inyong Ham 博士，Nabrasha-Lincon 大学的 Sang M. Lee 博士，Texas A&M 大学的 Way Kuo 博士，Wayne 州立大学的 Gary S. Wasserman 博士，Pittsburgh 大学的 Alice E. Smith 博士，Puerto Rico-Mayaguez 大学的 Gursel A. Suer 博士，浦项科技大学的 Moo Young Jung 博士，韩国科学技术院的黄鹤（Hawk Hwang）博士，中国科学院的徐伟宣博士和顾基发博士，中国东北大学的杨自厚博士，Thammasat 大学的 Suebsak Nanthavanij 博士，龙谷大学的人见胜人博士，大阪府立大学的田中英夫博士、太田宏博士、市桥秀友博士和石渊久生博士，名古屋大学的福田敏男博士和古桥武博士，广岛大学的坂和正敏博士，东京工业大学的小林重信博士和长尾智晴博士，岗山大学的大崎 一博士，东京都立大学的 Andrzey Osyczka 博士，东京都立科学技术大学的山崎源治博士，明治大学的向殿政男博士，庆广义塾大学的福川忠昭博士，大阪工业大学的和多田淳三博士，东海大学的中西祥八郎博士，电气通讯大学的松井正之博士，足利工业大学的小林敏孝博士、山城光雄博士、井田宪一博士、辻村泰宽博士和横田孝雄博士。

我们还特别感谢我们的博士研究生周根贵和龚弟金，他们仔细地校对了本书的手稿。我们还感谢足利工业大学的研究生：田口雄章、程春辉、李银珍、九保田爱丽卡、郑大伟和李银秀，在过去的几年里，他们和我们一起做了大量创造性的工作。

自然，我们还要感谢许许多多的研究者，本书的许多概念和观点都来自于他们的工

作。虽然不能一一提及他们的名字，我们希望能通过书后的参考文献表达出他们的贡献。

与 John Wiley & Sons 出版社的编辑和印刷人员一起工作的确是件愉快的事，特别要提到的是丛书的执行编辑 Robert L. Argentieri 先生，以及 Allison Ort Morvay 先生和 Minna Panfili 小姐。

我们还要感谢我们的妻子玄英子和张丽英，以及我们的孩子们，是他们的爱心、鼓励、理解和支持在我们写这本书时伴随我们度过了无数个夜晚和周末。

足利工业大学　玄光男

程润伟

1996年4月

目　录

第一章　遗传算法的基础

1.1 引　　言

工业工程中，特别是制造系统中的许多最优化问题性质非常复杂，很难用传统的优化方法来求解[458,470]。自1960年以来，人们对求解这类难解优化问题的兴趣日益增加。一种模仿生物自然进化过程的、被称为“进化算法(Evolutionary Algorithms)”的随机优化技术在解这类优化难题中显示出了通常优于传统优化算法的性能[11,373,437,453]。目前，进化算法主要包括三个研究领域：遗传算法、进化规划(Evolutionary Programming)和进化策略(Evolution Strategies)。其中，遗传算法是迄今为止进化算法中最广为人知的算法。

近年来，由于遗传算法求解复杂优化问题的巨大潜力及其在工业工程领域的成功应用，这种算法受到了广泛的关注。这些成功的应用包括：作业调度与排序、可靠性设计、车辆路径选择与调度、成组技术、设备布置与分配、交通问题，以及其他的许多问题。

1.1.1 遗传算法的一般结构

遗传算法的常用形式是Goldberg提出的[171]。遗传算法是一种基于生物自然选择与遗传机理的随机搜索算法。和传统搜索算法不同，遗传算法从一组随机产生的初始解，称为“种群(Population)”，开始搜索过程。种群中的每个个体是问题的一个解，称为“染色体(Chromosome)”。染色体是一串符号，比如一个二进制字符串。这些染色体在后续迭代中不断进化，称为遗传。在每一代中用“适值(Fitness)”来测量染色体的好坏[439]。生成的下一代染色体，称为后代(Offspring)。后代是由前一代染色体通过交叉(Crossover)或者变异(Mutation)运算形成的。新一代形成中，根据适值的大小选择部分后代，淘汰部分后代，从而保持种群大小是常数。适值高的染色体被选中的概率较高。这样，经过若干代之后，算法收敛于最好的染色体，它很可能就是问题的最优解或次优解。设$P(t)$和$C(t)$分别表示第t代的双亲和后代，遗传算法的一般结构可描述如下(并见图1.1)：

遗传算法过程

begin

$t \leftarrow 0$;

　初始化$P(t)$;

　评估$P(t)$;

　while 不满足终止条件 **do**

　begin

　　重组$P(t)$获得$C(t)$;

　　评估$C(t)$;

　　从$P(t)$和$C(t)$中选择$P(t+1)$;

$t \leftarrow t+1$;

end

end

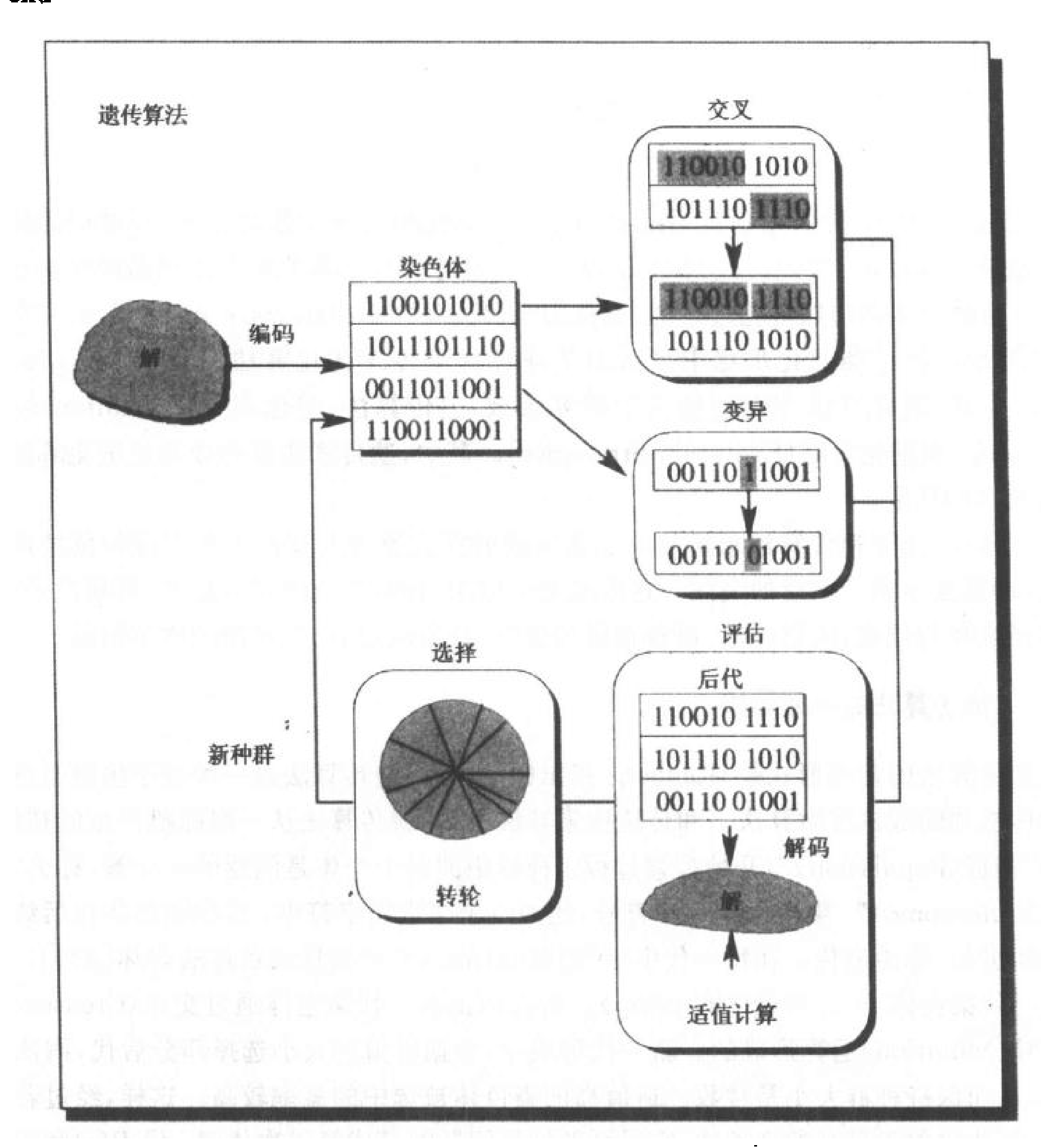


图 1.1　遗传算法的一般结构

以上是 Grefenstette 和 Baker 描述的改进的算法步骤[192,287]。通常初始化是随机产生的。重组包括交叉和变异来获得后代。实际上,遗传算法中有两类运算:

(1) 遗传运算:交叉和变异;

(2) 进化运算:选择。

遗传运算模拟了基因在每一代中创造新后代的繁殖过程,进化运算则是种群逐代更新的过程。以上描述和 Holland 的原始描述有些不同,原始描述是选择双亲来进行重组[220]。

交叉是最主要的遗传运算,它同时对两个染色体操作,组合二者的特性产生新的后代。交叉的最简单方法是在双亲(两个父辈的个体)的染色体上随机地选一个断点,将断点的右段互相交换,从而形成两个新的后代。这种方法对于二进制表示最为合适。遗传算法

的性能在很大程度上取决于采用的交叉运算的性能。

交叉率(记为 p_c)定义为各代中交叉产生的后代数与种群中的个体数(通常记为 *pop_size*)的比。它将参加交叉运算的染色体个数的期望值控制为 $p_c \times pop_size$。较高的交叉率可达到更大的解空间,从而降低停在非最优解上的机会;但是这个比率太高,会因搜索不必要的解空间而耗费大量的计算时间。

变异则是一种基本运算,它在染色体上自发地产生随机的变化。一种简单的变异方法是替换一个或多个基因。在遗传算法中,变异可以提供初始种群中不含有的基因,或找回选择过程中丢失的基因,为种群提供新的内容。

变异率(记为 p_m)定义为种群中变异基因数在总基因数中的百分比。变异率控制了新基因导入种群的比例。若变异率太低,一些有用的基因就不能进入选择;若太高,即随机的变化太多,那么后代就可能失去从双亲继承下来的好特性,这样算法就会失去从过去的搜索中学习的能力。

遗传算法在几个基本方面不同于传统优化方法。Goldberg 总结为如下几点[171]:

(1) 遗传算法运算的是解集的编码,而不是解集本身。

(2) 遗传算法的搜索始于解的一个种群,而不是单个解。

(3) 遗传算法只使用报酬信息(适值函数),而不使用导数或其他辅助知识。

(4) 遗传算法采用概率的,而不是确定的状态转移规则。

1.1.2 探索与扩展

搜索是人们对于问题的求解步骤没有先验知识时所采用的一种通用的方法。搜索可以采用盲目策略或者启发式策略[44]。盲目搜索策略不使用任何问题的领域信息;而启发式搜索则要使用把搜索引向最好方向的附加信息。搜索策略中包含两个重要方面,即探索最好解和扩展搜索空间[46]。Michalewicz 对爬山搜索、随机搜索和遗传搜索作了一个比较[287]。爬山法只注重探索最好解而忽略了搜索空间的扩展;随机搜索注重扩展空间却忽略了探索空间中最有希望的区域;而遗传算法是一类通用目的的搜索方法,它综合了定向搜索与随机搜索的优点,可以取得较好的区域探索与空间扩展的平衡。在遗传搜索的开始,随机的多样性的种群和交叉运算倾向于扩展搜索空间的大范围搜索,随着高适值解的获得,交叉运算倾向于在这些解的周围探索。换言之,交叉运算采用何种搜索策略(探索或扩展)取决于遗传系统的环境(种群的多样性)而不取决于运算本身。需要指出的是,简单的遗传运算设计为通用目的的搜索方法,它基本是一种盲目搜索而不能对后代进行改进。

1.1.3 基于种群的搜索

一般说来,优化算法是收敛于最优解的一系列计算步骤。多数经典的优化方法基于目标函数的梯度或高阶导数产生一个确定的计算系列。这类方法只作用于解空间中的单个解(参见图 1.2),随着迭代的进行,这个解沿着最速下降方向不断改进。这种点对点的搜索方法极可能陷入局部最优解。遗传算法通过保持一个潜在解的种群进行多方向搜索。这种种群对种群的搜索有能力跳出局部最优解。种群进行的是进化的模拟,每代中相对好的解可得到繁殖的机会,而相对差的解只得消亡。遗传算法采用概率转移律,以一定的概率选择部分个体繁殖,而令另一些个体消亡,从而将搜索引向解空间中最可能获得改进的

区域。

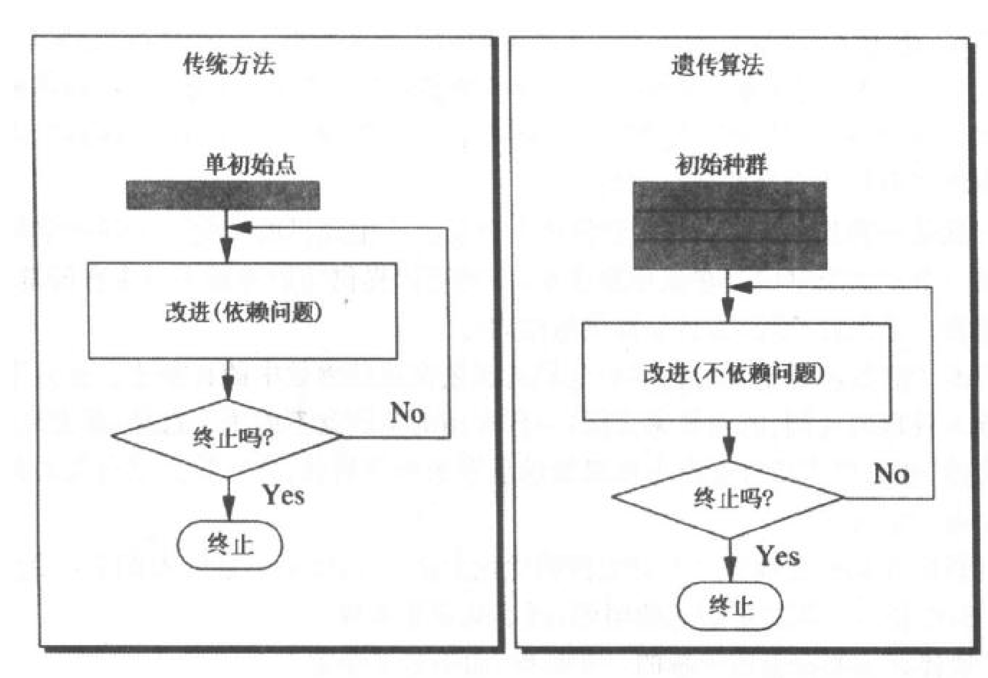


图 1.2 传统优化方法与遗传算法的比较

1.1.4 亚-启发式

最初,遗传算法是一种为许多难解问题设计的一种通用工具。早期的许多工作采用固定长度的二进制串作为通用的表达方式,遗传算子进行领域独立的二进制运算,而不使用任何字串所表达的领域知识。这种通用性反映了对具有广泛用途的鲁棒自适应系统设计的强调。然而,这种简单的遗传算法难以成功地直接用于许多难解的优化问题。对于各种特殊的问题,人们创造了各种以遗传算法为亚-启发式(Meta-Heuristics)的非标准的算法。Michalewicz 给他的著作起了个生动的名字:《遗传算法=数据结构+进化程序》,以强调我们不要局限于二进制的定长字串和算子,相反应多用自然表示(适于给定问题的数据结构)和能用于这种数据结构的有意义的遗传算子。

1.1.5 主要优点

作为一种新的优化技术,遗传算法受到了广泛的注意。它在解优化问题时有以下三大优点:

(1) 遗传算法对所解的优化问题没有太多的数学要求。由于它的进化特性,它在解的搜索中不需要了解问题的内在性质。遗传算法可以处理任意形式的目标函数和约束,无论是线性的还是非线性的,离散的还是连续的,甚至混合的搜索空间。

(2) 进化算子的各态历经性使得遗传算法能够非常有效地进行概率意义下的全局搜索,而传统的优化方法则是通过邻近点比较而移向较好点,从而达到收敛的局部搜索过程。这样,只有问题具有凸性时才能找到全局最优解,因为这时任何局优解都是全局最优解。

(3) 遗传算法对于各种特殊问题可以提供极大的灵活性来混合构造领域独立的启发式,从而保证算法的有效性。

1.1.6 遗传算法词汇

由于遗传算法起源于自然遗传学和计算机科学，因此遗传算法文献中的术语是自然与人工系统语言的混合。

在生物学中表明生物体构成的编码结构称为染色体，描述一个完整的生物体可能需要一种或多种染色体。一套完整的染色体称为基因型(Genotype)，而生成的生物体称为表现型(Phenotype)。每个染色体由一些称为基因的个体结构组成，每个基因表达了生物体的一种特性，基因在染色体结构中的分布，或称位置，决定了基因表达的特性。在各个特殊位置上一个基因可能具有几个表达不同特征的特殊值，基因的这些不同的值称为基因位值(Alleles)。

遗传算法的术语和最优化术语的对应关系见表 1.1。

表 1.1 遗传算法术语的说明

遗传算法	说明
染色体（位串，个体）	解（编码）
基因（位）	解的部分
位置	基因的分布
基因位值	基因的取值
表现型	解码后的解
基因型	编码表示的解

1.2 简单的遗传算法举例

本节我们用两个简单的例子详细说明遗传算法是如何工作的。这里采用的是 Michalewicz 给出的算法步骤[287]。

1.2.1 最优化问题

对于以下无约束优化问题：

$$\max f(x_1, x_2) = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2)$$
$$-3.0 \leqslant x_1 \leqslant 12.1$$
$$4.1 \leqslant x_2 \leqslant 5.8$$

目标函数的三维图形如图 1.3 所示。

1. *基因表达*

首先，将决策变量编码为二进制串，串长取决于需要的精度。例如，x_j 的值域是$[a_j, b_j]$，而要求的精度是小数点后 5 位，这就要求 x_j 的值域至少要分为$(b_j - a_j) \times 10^6$ 份。设 x_j 所需要的子串长为 m_j，则有

$$2^{m_j - 1} < (b_j - a_j) \times 10^6 \leqslant 2^{m_j} - 1$$

将 x_j 由二进制转为十进制可按下式计算：

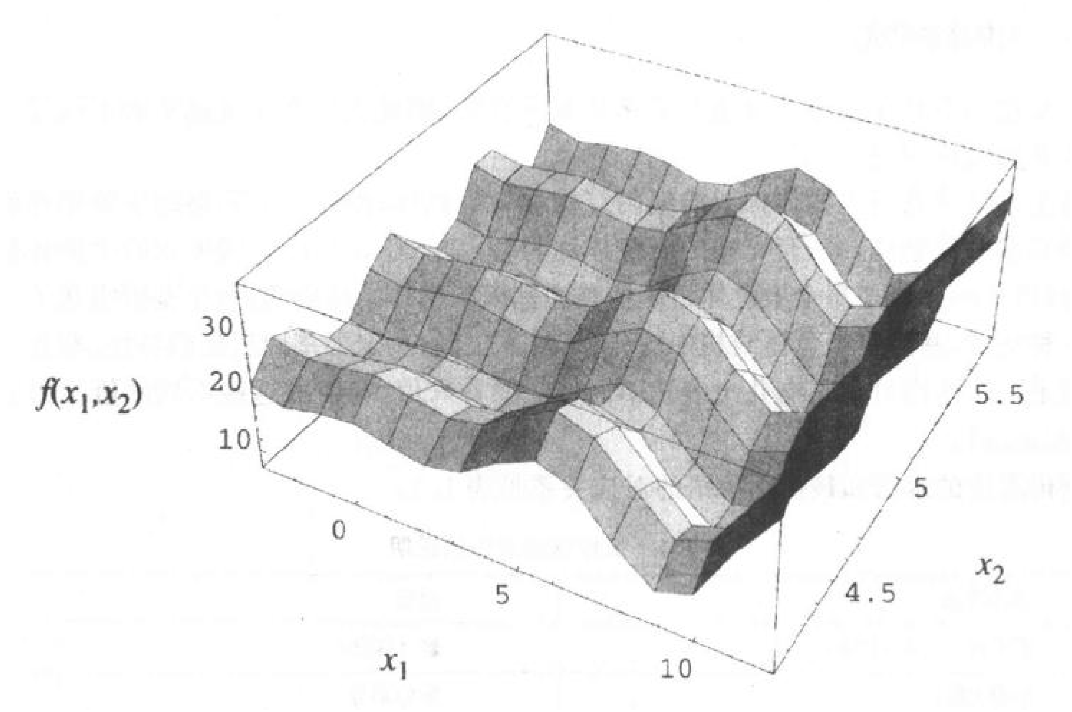


图 1.3 目标函数

$$x_j = a_j + decimal(substring_j) \times \frac{b_j - a_j}{2^{m_j} - 1}$$

$decimal(substring_j)$表示变量 x_j 的子串 $substring_j$ 的十进制值。

假设 x_1 和 x_2 需要的精度都是小数点后 5 位，两个变量需要的总串长按下述计算：

$$(12.1 - (-3.0)) \times 10\ 000 = 151\ 000$$

$$2^{17} < 151\ 000 \leqslant 2^{18}, \qquad m_1 = 18$$

$$(5.8 - 4.1) \times 10\ 000 = 17\ 000$$

$$2^{14} < 17\ 000 \leqslant 2^{15}, \qquad m_2 = 15$$

$$m = 18 + 15 = 33$$

染色体的总长是 33 位，可表示如下：

|←——33位——→|

$\mathbf{v}_j$ 000001010100101001 101111011111110

|←—18位—→| |←—15位—→|

对应的变量 x_1 和 x_2 的值为

	二进制码	十进制码
x_1	000001010100101001	5417
x_2	101111011111110	24318

十进制值为

$$x_1 = -3.0 + 5417 \times (12.1 - (-3.0)) / (2^{18} - 1) = -2.687969$$

$$x_2 = 4.1 + 24318 \times (5.8 - 4.1) / (2^{15} - 1) = 5.361653$$

2. 初始种群

随机产生的初始种群如下：

$\mathbf{v}_1 = [000001010100101001101111011111110]$

$\mathbf{v}_2 = [001110101110011000000010101001000]$

$\mathbf{v}_3 = [111000111000001000010101001000110]$

$\mathbf{v}_4 = [100110110100101101000000010111001]$

$\mathbf{v}_5 = [000010111101100010001110001101000]$

$\mathbf{v}_6 = [111110101011011000000010110011001]$

$\mathbf{v}_7 = [110100010011111000100110011101101]$

$\mathbf{v}_8 = [001011010100001100010110011001100]$

$\mathbf{v}_9 = [111110001011101100011101000111101]$

$\mathbf{v}_{10} = [111101001110101010000010101101010]$

对应的十进制值为

$\mathbf{v}_1 = [x_1, x_2] = [-2.687969, 5.361653]$

$\mathbf{v}_2 = [x_1, x_2] = [0.474101, 4.170144]$

$\mathbf{v}_3 = [x_1, x_2] = [10.419457, 4.661461]$

$\mathbf{v}_4 = [x_1, x_2] = [6.159951, 4.109598]$

$\mathbf{v}_5 = [x_1, x_2] = [-2.301286, 4.477282]$

$\mathbf{v}_6 = [x_1, x_2] = [11.788084, 4.174346]$

$\mathbf{v}_7 = [x_1, x_2] = [9.342067, 5.121702]$

$\mathbf{v}_8 = [x_1, x_2] = [-0.330256, 4.694977]$

$\mathbf{v}_9 = [x_1, x_2] = [11.671267, 4.873501]$

$\mathbf{v}_{10} = [x_1, x_2] = [11.446273, 4.171908]$

3. 评估

计算染色体适值的过程由以下三步构成：

估值过程

步骤 1：将染色体的基因型转换为表现型，即将二进制串转换为十进制值：$\mathbf{x}^k = (x_1^k, x_2^k)$，$k = 1, 2, \cdots, pop_size$. 这里，$pop_size$ 表示种群大小.

步骤 2：计算目标函数值 $f(\mathbf{x}^k)$.

步骤 3：将目标函数值转换为适值. 对于极大化问题，可简单地取目标值为适值：$eval(\mathbf{v}_k) = f(\mathbf{x}^k), k = 1, 2, \cdots, pop_size$.

适值函数起了测定染色体对目标的适应性的作用。

上例中染色体的适值函数值为

$eval(\mathbf{v}_1) = f(-2.687969, 5.361653) = 19.805119$

$eval(\mathbf{v}_2) = f(0.474101, 4.170144) = 17.370896$

$eval(\mathbf{v}_3) = f(10.419457, 4.661461) = 9.590546$

$$eval(v_4) = f(\ 6.159951, 4.109598) = 29.406122$$
$$eval(v_5) = f(-2.301286, 4.477282) = 15.686091$$
$$eval(v_6) = f(\ 11.788084, 4.174346) = 11.900541$$
$$eval(v_7) = f(\ 9.342067, 5.121702) = 17.958717$$
$$eval(v_8) = f(-0.330256, 4.694977) = 19.763190$$
$$eval(v_9) = f(\ 11.671267, 4.873501) = 26.401669$$
$$eval(v_{10}) = f(\ 11.446273, 4.171908) = 10.252480$$

显然，染色体 v_4 是最好的，而染色体 v_3 是最差的。

4. 选择

多数情况下采用转轮法作为选择方法，它是一种正比选择策略，能够根据与适值成正比的概率选出新的种群。转轮法由以下四步构成：

(1) 对各个染色体 v_k 计算适值 $eval(v_k)$

$$eval(v_k) = f(x);\quad k = 1,2,\cdots,pop_size$$

(2) 计算种群中所有染色体的适值的和

$$F = \sum_{k=1}^{pop_size} eval(v_k)$$

(3) 对各染色体 v_k，计算选择概率 p_k

$$p_k = \frac{eval(v_k)}{F};\quad k = 1,2,\cdots,pop_size$$

(4) 对各染色体 v_k，计算累积概率 q_k

$$q_k = \sum_{j=1}^{k} p_j;\quad k = 1,2,\cdots,pop_size$$

选择过程就是旋转转轮 *pop_size* 次，每次按如下方式选出一个染色体来构造新的种群：

选择过程

步骤 1：在[0,1]区间内产生一个均匀分布的伪随机数 r.

步骤 2：若 $r \leqslant q_1$，则选第一个染色体 v_1；否则，选择第 k 个染色体 $v_k (2 \leqslant k \leqslant pop_size)$，使得 $q_{k-1} < r \leqslant q_k$ 成立.

上例中，种群的适值和 F 为

$$F = \sum_{k=1}^{10} eval(v_k) = 178.135372$$

各染色体 $v_k (k=1,\cdots,10)$ 的选择概率 p_k 为

$$p_1 = 0.111180,\ p_2 = 0.097515,\ p_3 = 0.053839$$
$$p_4 = 0.165077,\ p_5 = 0.088057,\ p_6 = 0.066806$$
$$p_7 = 0.100815,\ p_8 = 0.110945,\ p_9 = 0.148211$$
$$p_{10} = 0.057554$$

各染色体的累积概率 q_k 为

$$q_1 = 0.111180,\ q_2 = 0.208695,\ q_3 = 0.262534$$

$q_4 = 0.427611$, $q_5 = 0.515668$, $q_6 = 0.582475$

$q_7 = 0.683290$, $q_8 = 0.794234$, $q_9 = 0.942446$

$q_{10} = 1.000000$

下面旋转转轮 10 次，每次选一个染色体来构造新种群. 设产生的[0,1]间的 10 个随机数序列如下：

0.301431，0.322062，0.766503，0.881893

0.350871，0.583392，0.177618，0.343242

0.032685，0.197577

第一个数 $r_1 = 0.301431$ 大于 q_3 小于 q_4，这表示染色体 v_4 被选来构造新种群；第二个数 $r_2 = 0.322062$ 也大于 q_3 小于 q_4，表示染色体 v_4 再次被新种群选中，重复以上操作，最后选出了如下新的种群：

$v'_1 =$ [100110110100101101000000010111001] (v_4)

$v'_2 =$ [100110110100101101000000010111001] (v_4)

$v'_3 =$ [001011010100001100010110011001100] (v_8)

$v'_4 =$ [111110001011101100011101000111101] (v_9)

$v'_5 =$ [100110110100101101000000010111001] (v_4)

$v'_6 =$ [110100010011111000100110011101101] (v_7)

$v'_7 =$ [001110101110011000000010101001000] (v_2)

$v'_8 =$ [100110110100101101000000010111001] (v_4)

$v'_9 =$ [000001010100101001101111011111110] (v_1)

$v'_{10} =$ [001110101110011000000010101001000] (v_2)

5. 交叉

本节采用单断点交叉法。该法随机地选择一个断点，交换双亲上断点的右端，生成新的后代。对于如下两个染色体，若随机断点选在第 17 个基因之后：

$v_1 =$ [100110110100101101000000010111001]

$v_2 =$ [001011010100001100010110011001100]

交换双亲上断点的右端后得到的两个后代如下：

$v'_1 =$ [100110110100101100010110011001100]

$v'_2 =$ [001011010100001101000000010111001]

设交叉率 $p_c = 0.25$，即平均有 25%染色体进行交叉，于是有：

交叉过程

begin

$k \leftarrow 0$;

while $k \leqslant 10$ **do**

```
    r_k ← [0,1]分布随机数;
      if r_k < 0.25 then;
        选 v_k 交叉的双亲之一;
    end
    k ← k + 1;
  end
end
```

设随机数序列如下：

0.625721，0.266823，0.288644，0.295114

0.163274，0.567461，0.085940，0.392865

0.770714，0.548656

于是染色体 v'_5 和 v'_7 被选入参加交叉。产生一个随机的整数 *pos* 作为断点，$pos \in [1,32]$（因为 33 是染色体的长度）。假设产生的 $pop = 1$，两染色体自第 1 位后切断，并交换断点右端，得到：

$$v'_5 = [100110110100101101000000010111001]$$
$$v'_7 = [000110110100101101000000010111001]$$
$$\downarrow$$
$$v'_5 = [100110110100101101000000010111001]$$
$$v'_7 = [000110110100101101000000010111001]$$

6. 变异

变异以等于变异率的概率改变一个或几个基因。假设 v_1 的第 16 个基因被选来变异，该基因为 1，故将其变为 0。于是变异后染色体为

$$v_1 = [100110110100101101000000010111001]$$
$$\downarrow$$
$$v'_1 = [100110110100101100000000010111001]$$

设变异率 $p_m = 0.01$，即种群中平均有 1% 基因发生变异。种群中共有 $m \times pop_size = 33 \times 10 = 330$ 个基因，这样每代平均有 3.3 个基因发生变异。为使每个基因有相同的机会发生变异，需要产生[0,1]间均匀分布的随机数序列 $r_k(k = 1, \cdots, 330)$。假设如下基因发生变异：

位址	染色体号	位号	随机数
105	4	6	0.009857
164	5	32	0.003113
199	7	1	0.000946
329	10	32	0.001282

变异后的新种群如下：

$$v_1' = [100110110100101101000000010111001]$$
$$v_2' = [100110110100101101000000010111001]$$
$$v_3' = [001011010100001100010110011001100]$$
$$v_4' = [111111001011101100011101000111101]$$
$$v_5' = [101110101110011000000010101001010]$$
$$v_6' = [110100010011111000100110011101101]$$
$$v_7' = [100110110100101101000000010111001]$$
$$v_8' = [100110110100101101000000010111001]$$
$$v_9' = [000001010100101001101111011111110]$$
$$v_{10}' = [001110101110011000000010101001010]$$

相应的十进制值和适值为

$$f(6.159951, 4.109598) = 29.406122$$
$$f(6.159951, 4.109598) = 29.406122$$
$$f(-0.330256, 4.694977) = 19.763190$$
$$f(11.907206, 4.873501) = 5.702781$$
$$f(8.024130, 4.170248) = 19.91025$$
$$f(9.342067, 5.121702) = 17.958717$$
$$f(6.159951, 4.109598) = 29.406122$$
$$f(6.159951, 4.109598) = 29.406122$$
$$f(-2.687969, 5.361653) = 19.805119$$
$$f(0.474101, 4.170248) = 17.370896$$

至此，我们完成了遗传算法的一次迭代。实验运行在 1 000 代后结束，在第 419 代就得到了最佳的染色体：

$$v^* = [111110000000111000111101001010110]$$
$$eval(v^*) = f(11.631407, 5.724824) = 38.818208$$
$$x_1^* = 11.631407$$
$$x_2^* = 5.724824$$
$$f(x_1^*, x_2^*) = 38.818208$$

1.2.2 配词问题

本例是 Freeman 1994 年给出的[143]，它很好地证明了遗传算法的效力。配词问题试图用遗传算法将随机产生的字母序列变为短语"to be or not to be"。由于短语中有 13 个字母，每个字母有 26 种可能，因此随机方式产生正确表达短语的概率是 $(1/26)^{13} = 4.03038\times10^{-19}$，即一千亿亿次中约有 4 次机会正确。我们用 ASCII 整数码来编码，英文

小写字母的 ASCII 码的范围为[97,122]。例如,字母串 *tobeornottobe* 转换为 ASCII 码为

[116, 111, 98, 101, 111, 114, 110, 111, 116, 116, 111, 98, 101]

产生一个有 10 个染色体的种群如下:

[114, 122, 102, 113, 100, 104, 117, 106, 97, 114, 100, 98, 101]
[110, 105, 101, 100, 119, 118, 121, 118, 106, 97, 104, 102, 106]
[115, 99, 121, 117, 101, 105, 115, 111, 115, 113, 118, 99, 98]
[102, 98, 102, 118, 114, 97, 109, 116, 101, 107, 117, 118, 115]
[107, 98, 117, 113, 114, 116, 106, 116, 106, 101, 110, 115, 98]
[102, 119, 121, 113, 121, 107, 107, 116, 122, 121, 111, 106, 104]
[116, 98, 120, 98, 108, 115, 111, 105, 122, 103, 103, 119, 109]
[101, 111, 111, 117, 114, 104, 100, 120, 98, 118, 116, 120, 97]
[100, 116, 114, 105, 117, 111, 115, 114, 103, 107, 109, 98, 103]
[106, 118, 112, 98, 103, 101, 109, 116, 112, 106, 97, 108, 113]

将它们转化为可读的形式为

rzfqdhujardbe
niedwvyvjahfj
scyueisosqvcb
fbfvramtekuvs
kbuqrtjtjensb
fwyqykktzyojh
tbxblsoizggwm
dtriuosrgkmbg
jvpbgemtpjalq

适值取为匹配的字母数。例如,字母串"rzfqdhujardbe"的适值是 2. 将变异改为以给定的概率取一个字母。用遗传算法运行 30 代后看看结果如何。每代中最好的染色体如表 1.2 所示。

表 1.2 每代中最好的字串

代数	字串	适值	代数	字串	适值
1	rzfqdhujardbe	2	8	rzfqohnottobe	8
2	rzfqdhuoardbe	3	9	rzfqohnottobe	8
3	rzfqghuoatdbe	4	10	rzfqohnottobe	8
4	rzfqghuoztobe	5	11	rzfqornottobe	9
5	rzfqghhottobe	6	12	rzfqornottobe	9
6	rzfqohhottobe	7	13	rwfwornottobe	9
7	rzfqohnottobe	8	14	rwcwornottobe	9

射)有三个主要的问题：

(1) 染色体的可行性；

(2) 染色体的合法性；

(3) 映射的唯一性。

可行性为染色体解码成为解之后是否在给定问题的可行域内的性质。合法性是指染色体是否代表了给定问题的一个解，如图 1.5 所示。

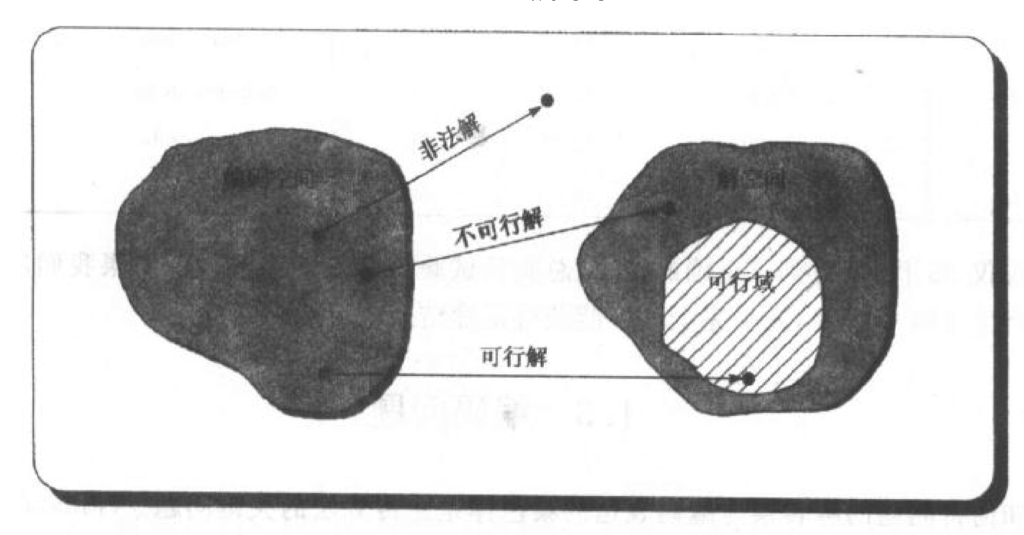


图 1.5 可行性与合法性

染色体的非可行性的概念源于约束优化问题，无论是传统方法还是遗传算法都必须满足约束。对于许多优化问题，可行域是用等式和不等式组表达的，这种情况下，许多有效的惩罚法可用来消除不可行的染色体[152,289,382]。在约束优化问题中，最优点通常位于可行域的边界上，惩罚法将迫使遗传搜索从可行域和不可行域两边同时逼近最优点。

非法性的概念源于编码技术。许多组合优化问题采用了问题专用的编码方法，这些编码方法采用单断点交叉通常可能获得非法的后代。由于非法的染色体不能解码为解，这样的染色体就不能进行评估，因此惩罚法就无法适用。通常采用的修复办法为将非法染色体转换为合法染色体。例如，著名的 PMX 算子(将在第三章介绍)就是为解决单断点交叉的非法性问题而提出的一种将替代编码和修复技术结合起来的双断点交叉方法。Orvosh 和 Davis 证明[322]，对于许多组合优化问题，修复非可行或非法染色体相对容易实现，这种修复策略远胜于拒绝策略和惩罚策略。

染色体到解的映射不外乎如下三种情况：

(1) 1-1 映射；

(2) n-1 映射；

(3) 1-n 映射。

如图 1.6 所示，1-1 映射是三种映射中最好的，而 1-n 映射是最差的。

为了构造一个有效的遗传算法，在设计新的非 0-1 串编码方法时必须仔细地考虑这些问题。

明了基于扩大的采样空间的选择过程。

以上方法的显著优点是可以用增加交叉率和变异率的方法来改进遗传算法的性能。如果选择是在扩大的采样空间中进行，就不用担心高的交叉率和变异率会造成太多的随机变动。

1.4.2 采样机理

采样机理是关于如何从采样空间中选择染色体的理论，选择染色体的方法有以下三种：

(1) 随机采样；

(2) 确定采样；

(3) 混合采样。

1. 随机采样

早期的工作大多采用随机采样方法。Baker 指出，这类方法的共同特点是在选择阶段根据生存概率来确定每个染色体的实际繁殖数[20]。选择阶段由两部分构成：

(1) 确定染色体的期望值；

(2) 将期望值转换为后代数。

染色体的期望值是表明该染色体应产生的后代的平均数的实数，采样过程则将这实数转换为后代数。

这类方法中最著名的是 Holland 正比选择或转轮选择，其基本思想是每个染色体的选择概率(即生存概率)正比于它的适值。对于适值为 f_k 的染色体 k，其选择概率 p_k 按下式计算：

$$p_k = f_k \Big/ \sum_{j=1}^{pop_size} f_j$$

然后，根据这些概率构造一个转轮。旋转转轮 *pop_size* 次，按上节介绍的方法每次选一个染色体加入新的种群。

Baker 提出的随机通用采样法只旋转一次转轮[20]。通用转轮按标准方式构造，其刻度数等于种群大小，染色体 k 的期望值 $e_k = pop_size \times p_k$。随机通用采样过程如下：

```
随机通用采样过程
begin
 sum ← 0;
  ptr ← rand();
  for k ← 1  to pop_size do
    sum ← sum + e_k;
    while sum > ptr do
      选择染色体 k;
      ptr ← ptr + 1;
    end
  end
```

相似的，相似性则是用简单的位与位相同的数量来测量的。注意，Holland 方法是选择双亲来组合，新的种群是由孩子替代掉双亲形成的。这称为繁殖计划(Reproductive Plan)。自 Grefenstette 和 Baker 的工作发表后[192]，选择过程变为按概率的机理来组成下一代。Michalewicz 详细描述了这种后代直接替代双亲，下代由转轮选择生成的简单遗传算法[287]。图 1.7 说明了基于规则采样空间的选择过程。

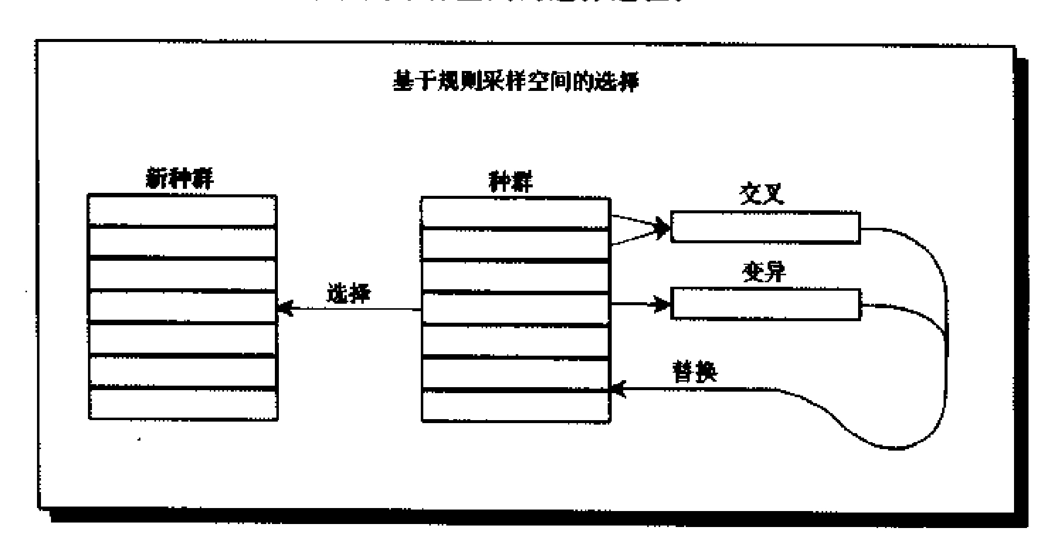


图 1.7　基于规则采样空间的选择

2. 扩大的采样空间

在扩大的采样空间中选择时，双亲和后代有同样的生存竞争机会，典型的例子是 $(\mu+\lambda)$-选择[130]。这种选择策略原用于进化策略中[373]，Bäck 和 Hoffmeister 将其引入到遗传算法中[10,13]。按这种策略，μ 个父代和 λ 个后代竞争生存，最后选出 μ 个最好的父代和后代构成下一代的双亲。另一种进化策略是(μ, λ)-选择，该策略选择 μ 个最好的后代作为下一代双亲 $(\mu<\lambda)$。两种方法都是确定性的，但可以转换成概率方法。虽然多数选择方法是基于规则采样空间的，但它们都可容易地在扩大采样空间中实现。图1.8说

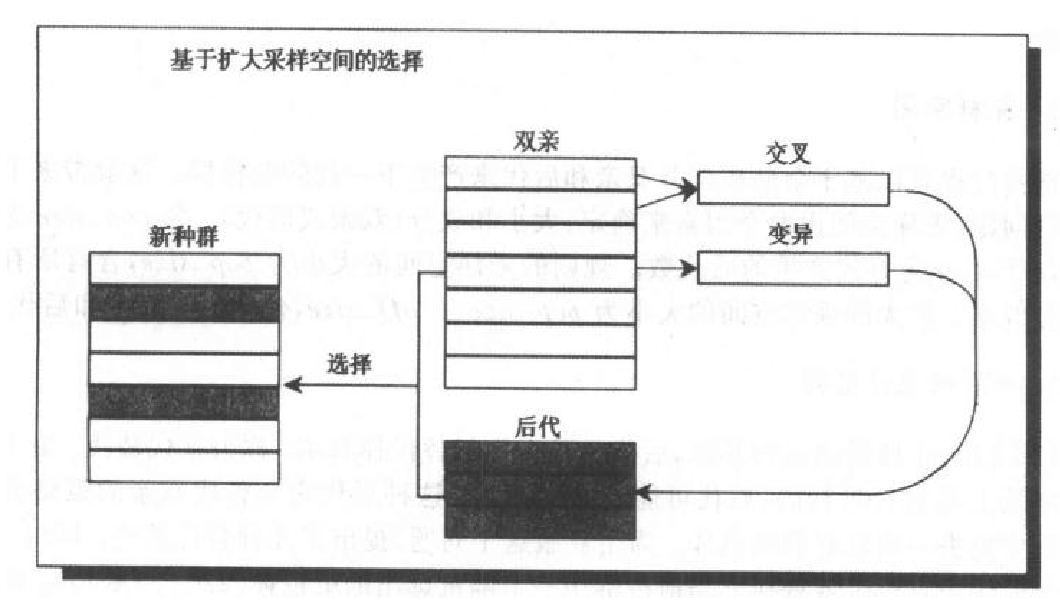


图 1.8　基于扩大采样空间的选择

明了基于扩大的采样空间的选择过程。

以上方法的显著优点是可以用增加交叉率和变异率的方法来改进遗传算法的性能。如果选择是在扩大的采样空间中进行，就不用担心高的交叉率和变异率会造成太多的随机变动。

1.4.2 采样机理

采样机理是关于如何从采样空间中选择染色体的理论，选择染色体的方法有以下三种：

(1) 随机采样；

(2) 确定采样；

(3) 混合采样。

1. 随机采样

早期的工作大多采用随机采样方法。Baker 指出，这类方法的共同特点是在选择阶段根据生存概率来确定每个染色体的实际繁殖数[20]。选择阶段由两部分构成：

(1) 确定染色体的期望值；

(2) 将期望值转换为后代数。

染色体的期望值是表明该染色体应产生的后代的平均数的实数，采样过程则将这实数转换为后代数。

这类方法中最著名的是 Holland 正比选择或转轮选择，其基本思想是每个染色体的选择概率(即生存概率)正比于它的适值。对于适值为 f_k 的染色体 k，其选择概率 p_k 按下式计算：

$$p_k = f_k \Big/ \sum_{j=1}^{pop_size} f_j$$

然后，根据这些概率构造一个转轮。旋转转轮 *pop_size* 次，按上节介绍的方法每次选一个染色体加入新的种群。

Baker 提出的随机通用采样法只旋转一次转轮[20]。通用转轮按标准方式构造，其刻度数等于种群大小，染色体 k 的期望值 $e_k = pop_size \times p_k$。随机通用采样过程如下：

```
随机通用采样过程
begin
 sum ← 0;
  ptr ← rand();
  for k ← 1  to pop_size do
    sum ← sum + e_k;
    while sum > ptr do
      选择染色体 k;
      ptr ← ptr + 1;
    end
  end
```

end

其中 $rand()$产生一个[0,1)均匀分布的伪随机数。

该方法的基本思想是保持每个染色体下一代的期望繁殖数。有趣的是有一种与此相反的方法禁止种群中的染色体加倍。其理由有两点:

(1) 阻止在种群中产生繁殖过多的超级染色体,因为这是快速收敛到局部最优(早熟)的主要原因。

(2) 保持种群的多样性,使得繁殖池中可为遗传搜索保留更多的信息。

存在的问题是删除多生出的染色体时,由保留的父代和后代形成的种群大小会小于预先指定的 *pop_size*。这时,通常采用初始化步骤来填补种群中的空缺。

2. 确定采样

确定采样是从采样空间中选择 *pop_size* 个最好的染色体,$(\mu+\lambda)$选择和 (μ,λ)选择都属于这种方法。Bäck 曾讨论过如何将它们转化为概率方法[10]。注意,这两种方法都是禁止染色体加倍的,所以多数研究者喜欢用其来解组合优化问题。

截断选择和阻塞选择也属于确定采样,它们将所有染色体按适值大小排序后,选择最好的作为繁殖的双亲。截断选择定义一个阈值 T,每代选 $T\%$的最好染色体,每个染色体产生 $100/T$ 个后代。阻塞选择等价于截断选择,它让选出的 *pop_size*/s 个最好染色体每个产生 s 个后代。当 $s=pop_size/T$ 时两者完全等价。

如果采用其他选择方法都未将最好的染色体选中,精英选择则可确保将最好的染色体留入下一代。

Brindle 的确定采样的基础是期望数的概念[48],即每个染色体的选择概率通常按公式 $p_k = f_k/\sum f_k$ 计算,而每个染色体的期望数为 $e_k = p_k \times pop_size$。首先,令每个染色体繁殖指标为期望值的整数部分,再将所有期望值的分数部分汇总起来,种群中不足的染色体由汇总表中分数值大的染色体来填充。

整代替代(后代替代所有双亲)也可视为一种确定选择。一种改进办法是用后代替换掉 n 个最坏的染色体,n 是产生的后代数。提出者 Whitely 将之称为 GENITOR[414],Syswerda 则称之为稳态繁殖[391]。Michalewicz 将之改为随机版,称为 modGA(改进遗传算法)[287],即后代是按生存概率来选择的,那些性能低于平均值的染色体会有较大的机会死去。

3. 混合采样

混合采样同时具有随机和确定采样的特征,典型的例子是 Goldberg 等提出的竞赛选择[173]。该方法随机地选出一组染色体,从中选出最好的一个进行繁殖,每组的染色体数称为竞赛人数。通常的竞赛人数为 2,称为 2 人竞赛。

随机竞赛选择是 Wetzel 提出的[413],这种方法按一般方法计算选择概率,用转轮法选出成对的染色体,然后将每对中适值高的染色体加入新种群,直到种群充满为止。

保留随机采样是 Brindle 提出的确定采样的改进版[48]。此法为每个染色体分配其期望值的整数部分个样本,然后所有染色体再按其期望值的分数部分来竞争种群中的空缺。

1.4.3 选择概率

本节讨论如何确定每个染色体的选择概率。按正比选择法，选择概率正比于染色体的适值。这种简单的方法有一些不好的性质，例如，在较早的代中一些超级染色体会霸占选择过程，而在较晚的代中种群集中在一起，染色体的竞争减弱，变得像随机搜索。

标定法与排序法就是为解决以上问题而提出的。标定法将原目标函数值映射为某个实数值，然后用这些实数值来确定每个染色体的生存概率。排序法则忽略实际目标函数值，而用染色体的顺序来替代生存概率。适值的标定具有双重用意：

(1) 使染色体之间的适值比保持合理的差距；

(2) 阻止某些超级染色体太快地把持遗传过程，以满足早期限制竞争、晚期鼓励竞争的需要。

对于多数标定法，标定参数是独立于问题的。适值排序与标定类似，但不需要额外的标定参数[348]。

自 De Jong 的工作之后，标定已成为广为接受的方法，几种不同的标定法已经提出。Goldberg [171] 和 Michalewicz [287] 对此作了极好的总结。一般说来，将染色体 k 的原始适值 f_k 转换为标定适值 f'_k 的过程可描述如下[192]：

$$f'_k = g(f_k)$$

其中，函数 $g(\cdot)$为转换函数，它可以取不同形式以获得不同的标定方法。例如，线性标定 σ 截法、幂律标定法、对数标定法等。这些方法可粗略地分为两类：

(1) 静态标定；

(2) 动态标定。

静态标定和动态标定按标定适值和原适值间的映射关系是恒定的还是可变的来划分。动态标定法进一步可分为以下两类：

(1) 标定参数按照每代中适值的分散情况自动调整，以保持恒定的选择压力；

(2) 标定参数随着代数增加而自动增加，以相应地增加选择压力。

注意，以上静态和动态标定的定义与 Bäck 和 Hoffmeister 给出的定义是不同的[13]。他们不是按标定适值和原适值，而是按选择概率和原适值之间的映射关系的可变性来定义的。

1. 线性标定

线性标定法调整所有染色体的适值，使得最好的染色体只能得到固定的、期望的数个后代，从而防止它过度繁殖。

当函数 $g(\cdot)$取线性形式时，有如下线性标定法：

$$f'_k = a \times f_k + b$$

这里，参数 a 和 b 的设定要使平均适值的染色体能产生一个后代，而最好染色体能产生指定的数个后代(通常指定为 2)。这种方法可能产生负的适值，负适值的染色体应让其死去。

2. 动态线性标定

当参数 b 随代数变化时，就有如下动态线性标定法：

$$f'_k = a \times f_k + b_t$$

b_t 的一种可取设定是取当前种群中的最小原适值，即 $b_t = -f_{min}$。

3. σ 截断

σ 截断是 Forrest 为改进线性标定法而提出的，以便能将问题的信息结合到映射中从而能适应有负适值问题的需要[134]。Goldberg 改进后的公式为[171]：

$$f'_k = f_k - (\bar{f} - c \times \sigma)$$

这里，c 是一个小整数；σ 为种群的标准差；$\bar{f}$ 是平均原适值。可能负的标定适值 f'_k 设为零。

选择压力是和种群中适值的分散度密切相关的。σ 截标定法利用了对适值的观察，并取适值均值与标准差作为标定因素。低于标准的染色体的适值设为零。这样就克服了由太坏的染色体引起的分散度过大的问题，从而减少了选择压力[206]。

4. 幂律标定

该法是 Gillies 提出的[169]，其中函数 $g(\cdot)$ 取原适值的指定幂的形式：

$$f'_k = f_k^{\alpha}$$

一般，α 根据问题确定，Gillies 取 $\alpha = 1.005$。

最好的与最坏的染色体标定适值的差随 α 值的增加而增加。当 α 趋近于零时，这个差也趋近于零，采样变为随机搜索。当 $\alpha > 1$ 时，差加大，采样大多分配给适值高的染色体。运行中 α 的值可能需要调整以扩大或缩小到需要的范围。

幂律标定的另一种定义为[192]

$$f'_k = (a \times f_k + b)^{\alpha}$$

5. 对数标定

该方法是 Fitzpatrick 和 Grefenstette 为映射极小化问题的目标函数而提出的[192]。$g(\cdot)$ 取为对数形式：

$$f'_k = b - \log(f_k)$$

其中，b 选择为大于任何 $\log(f_k)$ 的值。

6. 窗口技术

考虑 Hancock 讨论的情况[206]，给定适值分别为 2 和 1 的两个染色体，前者的后代数是后者的两倍。如果适值简单地加 1，两适值变为 3 和 2，而比成了 1.5。这说明简单地改变适值函数的基线就能有效地影响繁殖过程的速度。

窗口技术将移动基线(Moving Baseline)技术引入到适值正比选择中，以维持恒定的选择压力。该方法可视为一种动态线性标定法，取如下公式：

$$f'_k = f_k - f_w$$

其中，w 是窗口大小；典型的阶数是 2～10；f_w 是最后 w 代中获得的最差的值。

7. 正规化

正规化技术是 Cheng 和 Gen 提出的另一种动态标定技术[62]。对于极大化问题，它取如下形式：

$$f'_k = \frac{f_k - f_{\min} + \gamma}{f_{\max} - f_{\min} + \gamma}$$

这里，$f_{\max}$和 $f_{\min}$分别为当前种群中最好和最坏的原适值。该法可视为窗口大小 $w=1$ 的正规化的窗口法。γ 是一个小的正实数，通常限制在(0,1)开区间内。采用这种转换有双重目的：

(1) 防止方程中分母为零；

(2) 使适值-正比选择改为纯随机选择成为可能。

对于极小化问题，可取

$$f'_k = \frac{f_{\max} - f_k + \gamma}{f_{\max} - f_{\min} + \gamma}$$

8. Boltzmann 选择

Boltzmann 选择是一种用于正比选择的标定方法，它采用以下标定函数[10,287]：

$$f'_k = e^{f_k/T}$$

控制参数 T 大时，选择压力低。

9. 排序

Baker 将排序选择引入遗传算法，以克服直接基于适值的标定方法的缺点[17,172,206]。其基本思想为：将染色体从好到坏进行排序，按照它们在顺序中的位置而不是原适值指定选择概率。通常采用的两种方法为：线性排序和指数排序。

令 p_k 为种群中排在第 k 位的染色体的选择概率，线性排序取如下形式：

$$p_k = q - (k-1) \times r$$

其中，q 为最好染色体的选择概率，q_0 为最坏染色体的选择概率，参数 r 按下式确定：

$$r = \frac{q - q_0}{pop_size - 1}$$

中间的染色体按其位置成比例地从 q 减到 q_0。当设 $q_0=0$ 时，选择压力达到极大。

Michalewicz 提出如下指数排序法[287]：

$$p_k = q(1-q)^{k-1}$$

q 值越大，选择压力越大。Hancock 提出另一种指数排序法[206]：

$$p_k = q^{k-1}$$

这里，q 一般约为 0.99. 最好的染色体的标定适值接近于 1，而最坏的染色体为 q^{pop_size-1}。

1.4.4 选择压力

按照达尔文主义的观点，进化过程分为三大类[281]：

(1) 稳定选择；

(2) 定向选择；

(3) 破坏选择。

稳定选择又称正常选择，它倾向于消去具有极端值的染色体；定向选择具有增加或者减小种群均值的作用；而破坏选择则有删去中等值的染色体的趋势。多数选择方法都基于定向选择。

Kuo 和 Hwang 提出了一种基于破坏选择的方法[257]，这种方法采用非单调的适值函数，这和采用单调适值函数的传统方法大不相同。他们给出适值均值正规化函数的概念如下：

$$f_k' = | f_k - \bar{f} |$$

显然，该函数是非单调的。采用适值均值正规化函数的选择方法称为破坏选择，这种方法给最好和最坏的染色体以较高的选择概率。实验结果证明采用这种方法的遗传算法能够容易地找到"海底捞针"型函数的最优解。

Gen，Liu 和 Ida 提出了稳定选择法的一个例子[163]。按他们的方法，首先将染色体按原适值从大到小排序，然后定义三个偏好参数 p_1，p_0和 p_2($0<p_1<p_0<p_2<1$)，用以确定三个分别排在第 u_1，u_0 和 u_2 位的判断染色体，使得 $u_1 = \lfloor p_1 \cdot pop_size \rfloor$，$u_0 = \lfloor p_0 \cdot pop_size \rfloor$和 $u_2 = \lfloor p_2 \cdot pop_size \rfloor$。最后指定第 u_1 位的染色体的适值为 $e^{-1} \approx 0.37$，第 u_0 位的染色体的适值为 1，第 u_2 位的适值为 $2-e^{-1} \approx 1.63$。对于第 u 位的染色体 k 其指数适值 f_k 与位数 u 的关系为

$$f_k = \begin{cases} \exp\left[-\dfrac{u-u_0}{u_1-u_0}\right], & u < u_0 \\ 2-\exp\left[-\dfrac{u-u_0}{u_2-u_0}\right], & u \geqslant u_0 \end{cases} \tag{1.1}$$

如图 1.9 所示。

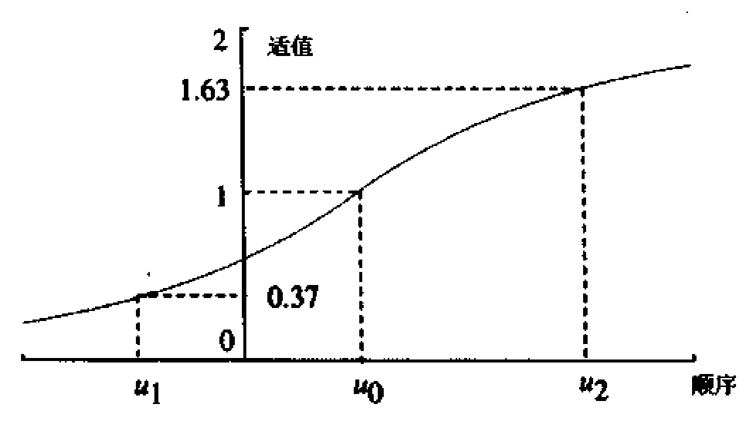


图 1.9 稳定标定

显然，偏好参数 p_1 和 p_2 是设计来删除具有极端值的染色体的。

Bäck 和 Hoffmeister 还给出了消亡选择和保存选择的概念[13]。保存选择让每个染色体都有非零的选择概率，而消亡选择则不是这样。消亡选择进一步分为左、右两种，左消

亡选择阻止最好的染色体繁殖，以避免由超级染色体产生的过早收敛；而右消亡选择则阻止最坏的染色体繁殖，以避免它们加大适值的分散性。Gen、Liu 和 Ida 提出的稳定标定法可以看作是左、右消亡选择的组合[163]，它能有效地阻止最好和最差的染色体的繁殖。

1.5 混合遗传算法

遗传算法是一种通用而有效的解最优化问题的方法，然而，单用简单的遗传算法在许多情况下不是十分有效，于是提出了多种混合算法。例如，Goldberg 运用 G-位的思想对二进制串的改进[171]；Davis 采用额外移动算子将专门领域的技术与遗传算法结合起来解决实际优化问题的方法[101]；Ackley 推荐的遗传爬山法，其中交叉似乎不起主导作用[2]；Reeves 的研究则将遗传算法视为一种广义的邻域搜索方法[349]；Mühlenbein 从理论上提出的[301]、由 Gorges-Schleuter [180] 实验证明的局部搜索可以起到关键的作用。Miller 等对于 *NP*-难的优化问题在纯的遗传算法中采用了局部改善运算[295]。

混合遗传算法的最常见的形式是在遗传算法的典型的重组选择循环中嵌入一块附加的局部优化模块。在这类混合方法中，对每个新产生的后代在其进入种群之前应用局部优化技术，使之移动到最近的局部最优点。当对染色体运用启发式方法作局部优化时，则采用遗传算法作全局最优点的探索。由于遗传算法与传统优化方法的互补性，混合算法通常比单一算法优越。已经发表了一些展现混合算法优越性的工作，其中有 Lamarck 的进化算法和元算法。

1.5.1 Lamarck 进化

将遗传算法和 Lamarck 进化理论结合起来的最早工作是，Grefenstette 将 Lamarck 算子引入遗传算法[190]；Davidor 定义了 Lamarck 变异概率，使得变异运算更易于控制，并改善了局部爬山的质量[93]；Shaefer 在染色体空间和解空间中增加了一个中间影映射，本质上就是 Lamarck 进化[377]。

Kennedy 对结合 Lamarck 进化理论的遗传算法作了说明[246，445]。Holland 的原始遗传算法起源于达尔文的自然选择理论，19 世纪，达尔文的理论受到了 Lamarck 的挑战，他提出环境变化通过有机生命体的变化引起生物结构的变化，并传给后代。以上理论认为生命体可以将它们获得的知识和经验传给后代。虽然，今天没有生物学家相信后天获得的能力能够遗传，但整个社会的进化符合 Lamarck 的理论。思想和知识通过结构化的语言和文化可以代代相传。遗传算法作为一个智能的机体可以从 Lamarck 的理论中获得益处，让一些个体把它的“经验”传给后代，就能提高遗传算法的寻优能力。按照 Lamarck 法，首先对最初的后代采用传统的爬山法使其快速达到局部最优，当它学到局域爬山的经验后，再让它进入评估与选择阶段。这些后代还有机会通过交叉将它们的经验传给后代。

令 $P(t)$ 和 $C(t)$ 为第 t 代的双亲和后代，混合遗传算法的一般结构可描述如下（参见图 1.10）：

混合遗传算法过程

begin

$t \leftarrow 0$;

初始化 $P(t)$；

评估 $P(t)$；

while 不满足终止条件 **do**

重组 $P(t)$以获得 $C(t)$；

局部爬山 $C(t)$；

评估 $C(t)$；

从 $P(t)$和 $C(t)$中选择 $P(t+1)$；

$t \leftarrow t+1$；

end

end

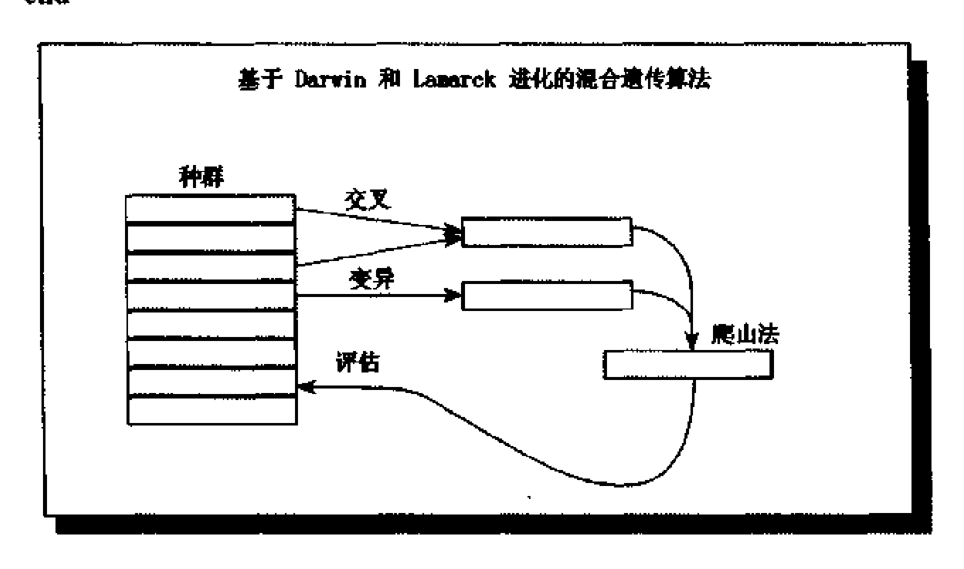


图 1.10 混合遗传算法的一般结构

在混合方法中，智能机体首先进行达尔文的生物进化，然后进行 Lamarck 的智能进化。传统的爬山法用在 Lamarck 进化中，以便在评估前将一些"智慧"注入到后代的机体中。

1.5.2 元算法

Moscato 和 Norman 提出用元算法(Memetic Algorithm)这一术语来描述一类局部搜索起主要作用的遗传算法[300]。这个术语来源于达尔文起的名词元(Meme)，元作为信息的单位在人们交换思想中被复制[103]。基因和元的主要差别是：元在传送之前，通常被传送者按照他的思想、理解和作法对元加以改造；而基因则是以整体形式来传递的。Moscato 和 Norman 将这种思想和局部改进结合起来，并建议用元算法来命名这类局部搜索较多的遗传算法。

Radcliffe 和 Surry 对元算法作了规范的描述[344]，提出了一个元算法和遗传算法的统一描述框架。按照 Radcliffe 和 Surry 的框架，如果在遗传算法中附加一个局部优化器，在每个后代进入种群前采用局优运算，那么元算法就可看作是局优子空间上的一类特殊的遗传搜索。重组和变异产生的解通常会在局优子空间之外，但是局部优化器可以修补这些解，使之产生的最后的后代在局优子空间内。得到的元算法如图 1.11 所示(改自

Radcliffe 和 Surry 的论文[344])。

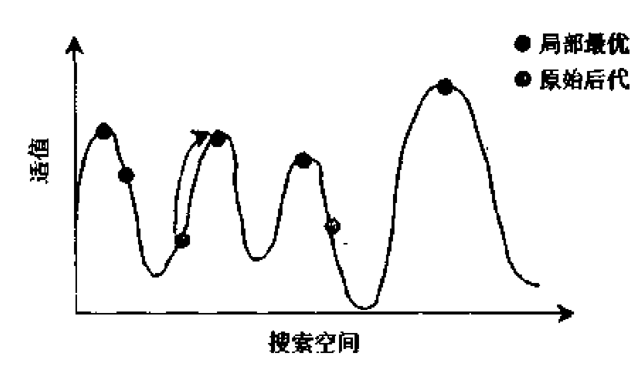


图 1.11 元算法与局部优化

局部搜索在遗传算法中已经引起人们极大的关注，许多成功的应用也支持了这类混合算法。元算法和 Lamark 进化都试图对基于不同自然现象的混合方法作出合理的解释。

1.6 遗传算法学界的重要事件

1.6.1 遗传算法的著作

遗传算法的最早雏形出现在 Fraser 的论文中，他是一位生物学家，试图仿真突出突变和选择交互作用的进化过程[139～142]。术语 epistasis 用来指明一个基因的突变对其他基因的影响。在遗传算法的这一领域里，这个术语用来表明基因位组合对染色体的适值的影响，它不是简单的各个基因位影响的线性函数[351]。

在 1975 年 Holland 的第一本关于遗传算法的书 *Adaptation in Natural and Artificial Systems* 和 De Jong 的博士论文 *An analysis of the behavior of a class of genetic adaptive systems*[104]出版之前，即使是知名科学家对遗传算法也不大了解[373]。Davis 说：Holland 开创了遗传算法这一领域，遗传算法的独特优点是由 Holland 和他的学生的精心而睿智的工作实现出来的[101]。在 Holland 的著作中，研究遗传算法的原动力是为可变的不确定的环境设计并实现鲁棒自适应系统。他的观点强调的是以从环境交互中获得的反馈函数来实现随时间变化的自适应系统的需要。这就产生了一些最初的再生或繁殖的方法，从而形成了今天称之为简单遗传算法的基础[106]。

此后，遗传算法覆盖了三个主要研究领域：基本遗传算法的研究，用遗传算法进行优化，和带有分类系统的机器学习[101]。这些研究的概况在 Goldberg 的著作 *Genetic Algorithms in Search, Optimization and Machine Learning* 中已有清楚的描述。

在过去 10 年中，遗传算法在实际中的应用大大增加。许多研究者开始对给定问题的搜索空间采用更自然的表达方式，这样就发展了一些适于新的数据结构的新遗传算法。于是，遗传算法的这些扩展和改进就给一般问题，特别是工业工程中难解的优化问题的求解带来了新的有希望的方向。这些问题定向的遗传算法和原来的基本遗传算法有很大的差别，这类遗传算法和其他进化算法的边界变得模糊不清了[11]。以上研究进展在 Michalewicz 的著作 *Genetic Algorithms+Data Structures=Evolution Programs* 中作了极好的讨论。

已经出版的遗传算法的著作见表 1.3，其他论文选集和会议文集见参考文献[436,440,447,450]。

表 1.3 遗传算法的著作

出版年份	作者	书名
1975	Holland[220]	*Adaptation in Natural and Artificial Systems*
1987	Davis[102]	*Genetic Algorithms and Simulated Annealing*
1989	Goldberg[171]	*Genetic Algorithms in Search, Optimization and Machine Learning*
1991	Davis[101]	*Handbook of Genetic* Algorithms
1991	Davidor[94]	*Genetic Algorithms and Robotics*
1992	Michalewicz[287]	*Genetic Algorithms + Data Structures = Evolution Programs*(2nd edition 1994, 3rd edition 1996)
1994	Bauer[25]	*Genetic Algorithms and Investment Strategies*
1994	Grefenstette[191]	*Genetic Algorithms for Machine Learning*
1994	Bhanu, Lee[33]	*Genetic Learning for Adaptive Image Segmentation*
1995	Chambers[54]	*Practical Handbook of Genetic Algorithms, Vols.* 1, 2
1995	Schwefel[373]	*Evolution and Optimum Seeking*
1996	Mitchell[296]	*An Introduction to Genetic Algorithms*
1996	Bäck[11]	*Evolutionary Algorithms in Theory and Practice*
1996	Lawton[204]	*A Practical Guide to Genetic Algorithms in C++*

1.6.2 学术会议与研讨会

自 1985 年以来，已经召开了多次遗传算法的学术会议和研讨会，为研究和应用的思想、进展和经验提供了国际交流的机会，并促进了理论与实际工作者之间的相互理解和合作。主要的会议见表 1.4。

表 1.4 遗传算法的学术会议

缩写	会议名称
ICGA	International Conference on Genetic Algorithms
PPSN	International Conference on Parallel Problem Solving from Nature
ICEC	IEEE International Conference on Evolutionary Computations
ANN&GA	International Conference on Artificial Neural Nets & Genetic Algorithms
EP	Annual Conference on Evolutionary Programming
FOGA	Workshop on Foundation of Genetic Algorithms
COGANN	International Workshop on Combinations of Genetic Algorithms and Neural Networks
EC	AISB Workshop on Evolutionary Computing

自 1985 年起，两年一次的系列会议 ICGA 开始召开，会议集中了对遗传算法理论与实践感兴趣的人(见表 1.5)。

表 1.5 ICGA 国际会议目录

年份	文集主编	会议地点
1985	Grefenstelle[186]	Pittsburgh, USA
1987	Grefenstelle[189]	Cambridge, USA
1989	Schaffer[370]	George Mason, USA
1991	Belew and Booker[30]	San Diego, USA
1993	Forrest[137]	Urbana-Champaign, USA
1995	Eshelman[120]	Pittsburgh, USA

两年一度的国际会议 PPSN 偶数年在欧洲召开，它是 ICGA 的姊妹会议。第一次会议是 1990 年在德国召开的。PPSN 会议的统一主题是自然计算，即源于自然的算法的设计、比较、理论与实验的理解，及其在科学与技术的实际问题中的应用(见表 1.6)。

表 1.6 PPSN 国际会议目录

年份	文集主编	会议地点
1990	Schwefel and Männer[374]	Dortmund, Germany
1992	Männer and Manderick[285]	Brussels, Belgium
1994	Davidor, Schwefel and Männer[95]	Jerusalem, Israel
1996	Ebeling and Voigt[438]	Dortmund, Germany

FOGA 研讨会始于 1990 年，两年召开一次。FOGA 偶数年召开，和奇数年召开的 ICGA 错开。两个会议都是在国际遗传算法学会(International Society for Genetic Algorithms)资助下组织和召开的。这个系列会议为以遗传算法的理论发表为目标的人提供了论坛(见表 1.7)。

表 1.7 FOGA 国际研讨会目录

年份	文集主编	会议地点
1990	Rawlins[347]	Bloomington, USA
1992	Whitley[416]	Colorado, USA
1994	Whitley and Vose[417]	Colorado, USA

1992 年，进化计算年会第一次会议在美国圣地亚哥召开。这个每年一次的会议是由进化规划学会(Evolutionary Programming Society)召开的。会议参加者可以欣赏到进化计算的广阔的领域，包括进化规划、进化策略、遗传算法、遗传规划，以及文化算法(Cultural Algorithms)等领域(见表 1.8)。

表 1.8 进化编程年会目录

年份	文集主编	会议地点
1992	Fogel and Atmar[133]	San Diego, USA
1993	Fogel and Atmar[134]	La Jolla, USA
1994	Sobald and Fogel[375]	San Diego, USA
1995	McDonnell, Reynolds and Fogel[284]	San Diego, USA
1996	Angeline and Bäck[435]	San Diego, USA

1993年，另一个两年一次的系列国际会议(International Conference on Artificial Neural Nets and Genetic Algorithms (ANN&GA))在奥地利召开。这个系列会议关心的主题是人工神经网络和遗传算法以及两者之间的交互(见表1.9)。

表1.9 ANN&GA会议目录

年份	文集主编	会议地点
1993	Albrech, Reeves and Steele[5]	Innsbruck, Austria
1995	Pearson, Steels and Albrecht[335]	Ales, France

另一个重要的年会是IEEE International Conference on Evolutionary Computation (ICEC),第一次会议是1994年在美国Orlando召开的。这个会议是IEEE神经网络学会主持的,包含了进化策略、进化规划、遗传算法和遗传规划等多个技术领域(见表1.10)。

表1.10 ICEC会议目录

年份	文集主编	会议地点
1994	Fogel[132]	Orlando, USA
1995	deSilva[108]	Perth, Australia
1996	Fogel[129]	Nagoya, Japan

AISB进化计算研讨会[128]于1994年在英国Leeds大学召开。这个研讨会是由人工智能与行为仿真研究学会(the Society for the Study of Artificial Intelligence and Simulation of Behavior)主持的，它为英国的从事进化计算的研究者提供了交流的机会，也是英国这一研究领域研究状况的最好反映。

1.6.3 遗传算法的杂志与杂志专集

杂志*Evolutionary Computation*(De Jong主编,MIT Press出版，1993开始发行）为遗传算法理论出版提供了论坛。MIT Press出版社的WWW网址为

http://www-mitpress.mit.edu

其中有一个分址提供订阅信息、每期的论文标题和摘要。

近年来,好几个杂志都出版了遗传算法和相关主题的专集,现列如下:

(1) D. B. Fogel and L. J. Fogel, guest editors, special issue on *Evolutionary Computation*, *IEEE Transactions on Neural Networks*, Vol. 5, No. 1, 1994.

(2) Z. Michalewicz, guest editor, special issue on *Evolutionary Computation*, *Statistics and Compting*, Vol. 4, No. 2, 1994.

(3) S. J. Raff, guest editor, special issue on *Genetic Algorithms*, *International Journal of Computers and Operations Research*, Vol. 22, No. 1, 1995.

(4) M. Gen, G. S. Wasserman and A. E. Smith, guest editors, special issue on *Genetic Algorithms and Industrial Engineering*, *International Journal of Computers and Industrial Engineering*, Vol. 30, No. 2, 1996.

(5) M. Gen and A. E. Smith, guest editors, special issue on *Intelligent Engineering*

Design, *International Journal of Engineering Design and Automation*, Vol. 2, No. 3, 1996.

(6) T. Ibaraki, guest editor, special issue on *Combinatorial Optimization*, *The Transactions of the Institute of Electrical Engineers of Japan*, Vol. 114-C, No. 4, 1994(in Japanese).

(7) J. Watada, editor-in-chief, special issue on *Fuzzy and Genetic Algorithms*, *Journal of Japan Society for Fuzzy Theory and Systems*, Vol. 7, No. 5, 1995 (in Japanese).

(8) H. Hirata and J. Miyamichi, guest editors, special issue on *New Approach to Combinatorial Problems and Scheduling Problems*, *Transactions of the Society of Instrument and Control Engineers of Japan*, Vol. 31, No. 5, 1995 (in Japanese).

1.6.4 互联网上公共访问的遗传算法信息

随着互联网的普及，网上有许多遗传算法的节点对公众开放，可通过匿名的FTP或WWW访问。其中,GA Archives是最著名的,可以按以下网址进入：

http://www.aic.nrl.navy.mil/galist/

GA Archive是遗传算法研究信息的总库,从中可查到遗传算法文摘,实现遗传算法的原程序和相关会议的通知,和全世界包含进化计算材料的许多网点联结的通道。这个网点是由美国海军人工智能应用研究中心维护的。

主要内容有：

(1) 进化计算相关事件日历；

(2) 遗传算法成就目录：期刊，原程序,信息；

(3) 其他遗传算法相关信息的通道；

(4) 进化计算研究组的通道。

在互联网上遗传算法的另一重要资源是ENCORE (the EvolutioNary COmputation REpository network)。ENCORE有一个文件夹,即进化计算领域相关电子资源的汇编。它分为几个大类,反映了现行的主要研究方向:遗传算法、进化规划、遗传规划、分类机系统。

ENCORE不在一个单一网点上,而是一个含有同样信息的服务网。读者可以选用最近的EClair (即ENCORE网上的网点)。目录如下：

The EClair at EUnet Deutschland GmbH：
  ftp://ftp.Germany.EU.net/pub/research/softcomp/EC/

The EClair at the Santa Fe Institutte：
  ftp://alife.santafe.edu/pub/USER-AREA/EC/

The EClair at the Chinese University of Hong Kong：
  ftp://ftp.dcs.warwick.ac.uk/pub/mirrors/EC/

The EClair at the California Institute of Technology：
  ftp://ftp.krl.caltech.edu/pub/EC/

The EClair at Wayne State University, Detroit:

ftp://ftp.cs.wayne.edu/pub/EC/

The EClair at the Michigan State University:

ftp://ftp.egr.msu.edu/pub/EC/

The EClair at the University of Capetown, South Africa:

ftp://ftp.uct.ac.za/pub/mirrors/EC/

The EClair at Technical University of Berlin, Germany:

ftp://ftp-bionik,fb10.tu-berlin.de/pub/EC/

The EClair at Purdue University, West Lafayette, USA:

ftp://coast.cs.purdue.edu/pub/EC/

The EClair at the University of Oviedo, Spain:

ftp://zeus.etsimo.uniovi.es/pub/EC/

The EClair at CEFET-PR, Curitiba, Brazil:

ftp://ftp.cefetpr.br/pub/EC/

第二章　约束优化问题

最优化在运筹学和管理科学中起着核心作用。最优化通常是极大或极小某个多变量的函数并满足一些等式和/或不等式约束。最优化技术对社会的影响日益增加，应用的种类和数量快速增加，丝毫没有减缓的趋势。

然而，许多工业工程设计问题性质十分复杂，用传统的优化方法很难解决。近年来，遗传算法作为一种全新的优化方法，以其巨大的潜力受到人们的普遍关注。本章将讨论遗传算法在无约束规划、非线性规划、随机规划、目标规划和区间规划中的应用。

2.1　无约束优化

无约束优化针对的是没有任何约束前提下的极大或极小化某个函数的问题[28]。一般，无约束优化问题可用如下数学公式描述[279]：

$$\begin{aligned} &\min \quad f(\mathbf{x}) \\ &\text{s.t.} \quad \mathbf{x} \in \Omega \end{aligned}$$

其中，f 是实值函数，可行域 Ω 是 E^n 的子集。就可行域而言，当 $\Omega = E^n$ 时，问题是完全无约束的。但本节只考虑 Ω 是 E^n 中几个特殊子集的情况，我们把注意力集中在遗传算法的求解技术上。

点 $\mathbf{x}^* \in \Omega$，是 Ω 上的 f 的局部最优点，如果存在 $\varepsilon > 0$ 使得所有 $\mathbf{x} \in \Omega$ 与 $\mathbf{x}^*$ 的距离不大于 ε 的点满足 $f(\mathbf{x}) \geqslant f(\mathbf{x}^*)$。点 $\mathbf{x}^* \in \Omega$，是 f 在 Ω 上的全局最优点，如果 $f(\mathbf{x}) \geqslant f(\mathbf{x}^*)$ 对所有 $\mathbf{x} \in \Omega$ 成立。以上局部最优的必要条件是基于 f 的微分得出的。f 的梯度定义如下：

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \cdots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right]$$

f 在点 $\mathbf{x}$ 的 Hessian 矩阵记为 $\nabla^2 f(\mathbf{x})$ 或 $\mathbf{F}(\mathbf{x})$：

$$\mathbf{F}(\mathbf{x}) = \left[\frac{\partial^2 f(\mathbf{x})}{\partial x_i x_j}\right]$$

虽然绝大多数实际优化问题都有必须满足的约束，但无约束优化问题的研究是约束优化问题的基础。本节讨论如何用遗传算法来解无约束优化问题。下一节就可自然地扩展本节的方法，为约束优化提供求解方法。

2.1.1　Ackley 函数

Ackley 函数是指数函数叠加上适度放大的余弦波再经调制而得到的连续型实验函数[2,11]。它的拓扑形状如图 2.1 所示，其特征是在一个几乎平坦的区域内由余弦波调制形成一个个孔或峰，从而使曲面起伏不平。

Ackley 函数如下：

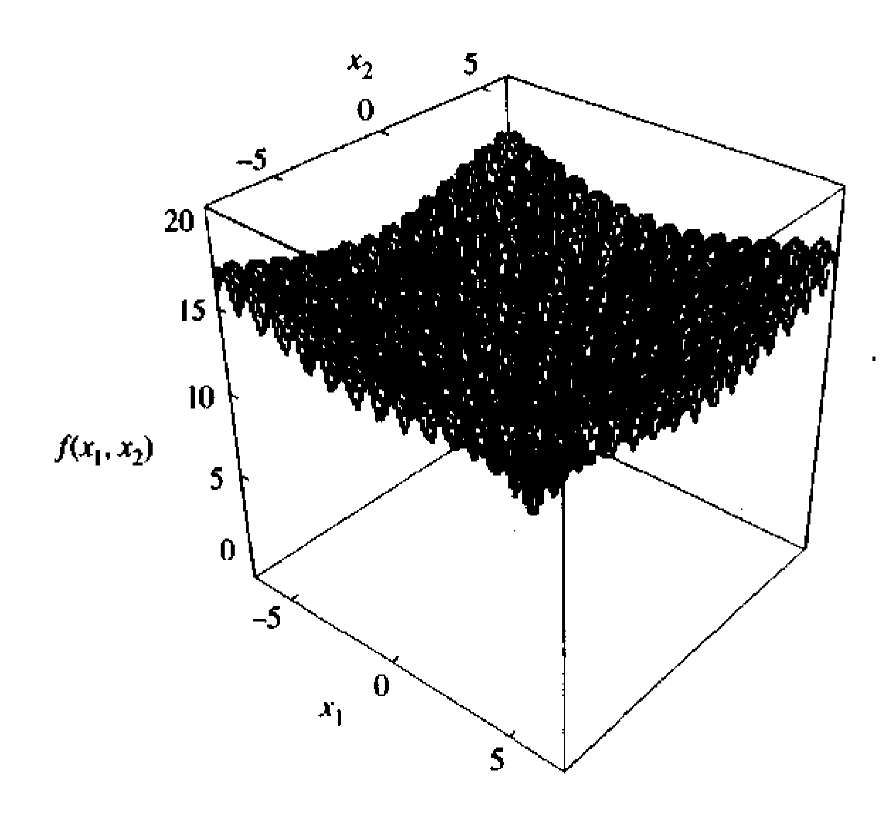


图 2.1 Ackley 函数

$$\min f(x_1, x_2) = -c_1 \cdot \exp\left(-c_2\sqrt{\frac{1}{n}\sum_{j=1}^{2} x_j^2}\right) - \exp\left(\frac{1}{n}\sum_{j=1}^{2}\cos(c_3 \cdot x_j)\right) + c_1 + e$$

$$-5 \leqslant x_j \leqslant 5;\ j = 1,2$$

这里，$c_1=20$，$c_2=0.2$，$c_3=2\pi$，$e=2.71282$。并且最优解为$(x_1^*, x_2^*)=(0,0)$，$f(x_1^*, x_2^*)=0$。

Ackley 指出，这个函数的搜索十分复杂，因为一个严格的局优算法在爬山过程中不可避免地要落入局部最优的陷阱，而扫描较大邻域就能越过干扰的山谷，逐步达到较好的最优点。所以，Ackley 函数为遗传算法提供了一个有力的例证。

2.1.2 用遗传算法极小化 Ackley 函数

为极小化 Ackley 函数，我们简单地采用如下遗传算法来实现：

(1) 实数编码；

(2) 算术交叉；

(3) 非均匀变异；

(4) 最好种群 *pop_size* 选择。

算术交叉定义为两个染色体的如下组合方式[287]：

$$\mathbf{v}_1' = \mathbf{v}_1 + (1-\lambda)\,\mathbf{v}_2$$

$$\mathbf{v}_2' = \mathbf{v}_2 + (1-\lambda)\,\mathbf{v}_1$$

这里，$\lambda \in (0,1)$。

非均匀变异如下[287]：对于给定的父亲 $\mathbf{v}$，若他的元素 x_k 被选来变异，则生成的后

代为 $\mathbf{v}' = [x_1, \cdots, x_k', \cdots, x_n]$，其中，$x_k'$ 随机地按如下两种可能的机会变化：

$$x_k' = x_k + \Delta(t, x_k^U - x_k)$$

或

$$x_k' = x_k - \Delta(t, x_k - x_k^L)$$

这里，x_k^U 和 x_k^L 分别是 x_k 的上下界。函数 $\Delta(t,y)$ 给出 $[0,y]$ 间的一个值，使得 $\Delta(t,y)$ 随着 t 的增加而趋于 0（t 是代数）。如可取 $\Delta(t,y)$ 为

$$\Delta(t,y) = y \cdot r \cdot \left(1 - \frac{t}{T}\right)^b$$

这里，r 是[0,1]间的随机数；T 最大代数；b 是确定非均匀度的参数。

最好种群选择是从父代和后代中为下一代选择最好的 *pop_size* 个染色体。对于本例可取目标值作为适值，并按适值选取染色体。

遗传算法的参数设定如下：

种群大小：*pop_size*=10

最大代数：*max_gen*=1 000

变异率：p_m=0.1

交叉率：p_c=0.3

初始种群是在[−5,5]间随机产生的：

	x_1	x_2
$\mathbf{v}_1$ =	[4.954222,	0.169225]
$\mathbf{v}_2$ =	[−4.806207,	−1.630757]
$\mathbf{v}_3$ =	[4.672536,	−1.867275]
$\mathbf{v}_4$ =	[1.897794,	−0.196387]
$\mathbf{v}_5$ =	[−2.127598,	0.750603]
$\mathbf{v}_6$ =	[−3.832667,	−0.959655]
$\mathbf{v}_7$ =	[−3.792383,	4.064608]
$\mathbf{v}_8$ =	[1.182745,	−4.712821]
$\mathbf{v}_9$ =	[3.812220,	−3.441115]
$\mathbf{v}_{10}$ =	[−4.515976,	4.539171]

相应的适值为

$eval(\mathbf{v}_1)=f(\ 4.954222,\quad 0.169225)=10.731945$

$eval(\mathbf{v}_2)=f(-4.806207,\quad -1.630757)=12.110259$

$eval(\mathbf{v}_3)=f(\ 4.672536,\quad -1.867275)=11.788221$

$eval(\mathbf{v}_4)=f(\ 1.897794,\quad -0.196387)=\ 5.681900$

$eval(\mathbf{v}_5)=f(-2.127598,\quad 0.750603)=\ 6.757691$

$eval(\mathbf{v}_6)=f(-3.832667,\quad -0.959655)=\ 9.194728$

$eval(\mathbf{v}_7)=f(-3.792383,\quad 4.064608)=11.795402$

$eval(\mathbf{v}_8)=f(\ 1.182745,\quad -4.712821)=11.559363$

$eval(\mathbf{v}_9)=f(\ 3.812220,\quad -3.441115)=12.279653$

$$eval(\mathbf{v}_{10})=f(-4.515976,\quad 4.539171)=14.251764$$

下面对这些染色体作交叉运算。首先产生随机数系列：

0.828211， 0.199683， 0.639149， 0.629170， 0.957427

0.149358， 0.304788， 0.058504， 0.149693， 0.326670

这表明染色体 $\mathbf{v}_2$,$\mathbf{v}_6$,$\mathbf{v}_8$ 和 $\mathbf{v}_9$ 被选出参加交叉。产生的后代为

$$\mathbf{v}_2=[-4.806207,-1.630757]$$
$$\mathbf{v}_6=[-3.832667,-0.959655]$$
$$\downarrow$$
$$\mathbf{v}_1'=[-4.444387,-1.383817]$$
$$\mathbf{v}_2'=[-4.194488,-1.206594]$$

和

$$\mathbf{v}_8=[\ 1.182745,-4.712821]$$
$$\mathbf{v}_9=[\ 3.812220,-3.441115]$$
$$\downarrow$$
$$\mathbf{v}_3'=[\ 3.683262,-4.521950]$$
$$\mathbf{v}_4'=[\ 1.311703,-3.631985]$$

然后作变异.由于种群中总共有 $2\times10=20$ 个基因,因此应产生[0,1]间的随机数系列 $r_k(k=1,\cdots,20)$。作变异的相应基因为

位	染色体	变量	随机数
11	6	x_1	0.081393

生成的后代为

$$\mathbf{v}_5'=[-4.068506,-0.959655]$$

各个后代的适值如下：

$$eval(\mathbf{v}_1')=f(-4.444387,\quad -1.383817)=11.927451$$
$$eval(\mathbf{v}_2')=f(-4.194488,\quad -1.206594)=10.566867$$
$$eval(\mathbf{v}_3')=f(\ 3.683262,\quad -4.521950)=13.449167$$
$$eval(\mathbf{v}_4')=f(\ 1.311703,\quad -3.631985)=10.538330$$
$$eval(\mathbf{v}_5')=f(-4.068506,\quad -0.959655)=9.083240$$

从双亲与后代中选择 10 个最好的染色体组成新的种群：

	x_1	x_2
$\mathbf{v}_1$=[	4.954222,	0.169225]
$\mathbf{v}_2$=[	1.311703,	−3.631985]
$\mathbf{v}_3$=[	4.672536,	−1.867275]
$\mathbf{v}_4$=[	1.897794,	−0.196387]
$\mathbf{v}_5$=[	−2.127598,	0.750603]
$\mathbf{v}_6$=[	−3.832667,	−0.959655]
$\mathbf{v}_7$=[	−3.792383,	4.064608]
$\mathbf{v}_8$=[	1.182745,	−4.712821]

$$\mathbf{v}_9=[-4.194488, \quad -1.206594]$$
$$\mathbf{v}_{10}=[-4.068506, \quad -0.959655]$$

相应的适值为

$$eval(\mathbf{v}_1)=f(4.954222, \quad 0.169225)=10.731945$$
$$eval(\mathbf{v}_2)=f(1.311703, \quad -3.631985)=10.538330$$
$$eval(\mathbf{v}_3)=f(4.672536, \quad -1.867275)=11.788221$$
$$eval(\mathbf{v}_4)=f(1.897794, \quad -0.196387)=5.681900$$
$$eval(\mathbf{v}_5)=f(-2.127598, \quad 0.750603)=6.757691$$
$$eval(\mathbf{v}_6)=f(-3.832667, \quad -0.959655)=9.194728$$
$$eval(\mathbf{v}_7)=f(-3.792383, \quad 4.064608)=11.795402$$
$$eval(\mathbf{v}_8)=f(1.182745, \quad -4.712821)=11.559363$$
$$eval(\mathbf{v}_9)=f(-4.194488, \quad -1.206594)=10.566867$$
$$eval(\mathbf{v}_{10})=f(-4.068506, \quad -0.959655)=9.083240$$

至此，我们仅完成了一次遗传迭代(繁殖一代)。在第 1 000 代时，得到了如下染色体：

$$\begin{array}{rll} & x_1 & x_2 \\ \mathbf{v}_1= & [-0.000002, & -0.000000] \\ \mathbf{v}_2= & [-0.000002, & -0.000000] \\ \mathbf{v}_3= & [-0.000002, & -0.000000] \\ \mathbf{v}_4= & [-0.000002, & -0.000000] \\ \mathbf{v}_5= & [-0.000002, & -0.000000] \\ \mathbf{v}_6= & [-0.000002, & -0.000000] \\ \mathbf{v}_7= & [-0.000002, & 0.000000] \\ \mathbf{v}_8= & [-0.000002, & -0.000000] \\ \mathbf{v}_9= & [-0.000002, & -0.000000] \\ \mathbf{v}_{10}= & [-0.000002, & -0.000000] \end{array}$$

适值为 $f(x_1^*, x_2^*) = -0.005456$。

2.2 非线性规划

非线性规划是对存在等式和/或不等式约束的前提下最优化某个目标函数的问题[452,454,465]。由于许多实际问题不能成功地表述为线性规划模型，因此，非线性规划对于工程、数学和运筹学的各个领域都是极其重要的工具。一般非线性规划可描述如下：

$$\max \quad f(\mathbf{x}) \tag{2.1}$$
$$\text{s.t.} \quad g_i(\mathbf{x}) \leqslant 0; \quad i = 1,2,\cdots,m_1 \tag{2.2}$$
$$h_i(\mathbf{x}) = 0; \quad i = m_1 + 1,\cdots, m(= m_1 + m_2) \tag{2.3}$$
$$\mathbf{x} \in X \tag{2.4}$$

这里 $f;g_1,g_2,\cdots,g_{m_1};h_{m_1+1},h_{m_1+2},\cdots,h_m$ 都是定义在 E^n 上的实值函数。X 是 E^n 上的子集，$\mathbf{x}$ 是 n 维实向量，其分量为 $x_1,x_2,\cdots,x_n$。上述问题要求在变量 $x_1,x_2,\cdots,x_n$ 满足约束

的同时极小化函数 f。函数 f 通常称为目标函数，或判据函数。约束 $g_i(\mathbf{x}) \leqslant 0$ 称为不等式约束；约束 $h_i(\mathbf{x}) = 0$ 称为等式约束。集合 X 通常为变量的上下界限定的区域。向量 $\mathbf{x} \in X$ 且满足所有约束，则称之为问题的可行解。所有可行解的集合组成可行域。非线性规划问题是要找一个可行解 $\bar{\mathbf{x}}$ 使得 $f(\mathbf{x}) \leqslant f(\bar{\mathbf{x}})$ 对于所有可行解 $\mathbf{x}$ 成立。那么，$\bar{\mathbf{x}}$ 为最优解。和线性规划不同，传统的非线性规划方法十分复杂且效率不高。过去几年里，用遗传算法解非线性规划的工作不断增加[354]。本节说明如何用遗传算法来解非线性规划问题。

2.2.1 满足约束

由于对染色体作遗传运算时通常获得不可行的后代，因此运用遗传算法解非线性规划的核心问题是如何满足约束的问题。近年来，已经提出了几种用遗传算法满足约束的技术[248,288,308,322]。Michalewicz 对此作了一个极好的综述[289,291]。这些技术大致可分为以下几类：

(1) 拒绝策略；

(2) 修复策略；

(3) 改进遗传算子策略；

(4) 惩罚策略。

各种策略都有不同的优点和缺点。

1. 拒绝策略

拒绝策略抛弃所有进化过程中产生的不可行的染色体。这是遗传算法中普遍的作法。当可行的搜索空间是凸的，且为整个搜索空间的适当的一部分时，这种方法应该是有效的。然而，这是很严格的限制。例如，对许多约束优化问题初始种群可能全由非可行染色体构成，这就需要对它们进行修补。对于某些系统(特别是可行搜索空间非凸时)，允许跨过不可行域时修复往往更容易达到最优解。

2. 修复策略

修补染色体是对不可行染色体采用修复程序使之变为可行的。对于许多组合优化问题，构造修复程序相对比较容易。Liepins 和他的合作者通过对遗传算法的性能实验测试证明，对于一个有多个不连通可行集的约束组合优化问题，修复策略在速度和计算性能上都远胜过其他策略[274,275]。

修复策略取决于是否存在一个可将不可行后代转化为可行的修复程序。该方法的缺点是它对问题本身的依赖性，对于每个具体问题必须设计专门的修复程序。对于某些问题，修复过程甚至比原问题的求解更复杂。

修复后的染色体可以只用来作评估，也可用来替代原染色体进入种群。Liepins 等采用永不替代法，即不让修复过的染色体进入种群[274,275]；而 Nakano 和 Yamada 采用了始终替代法[313]。最近，Orvosh 和 Davis 提出了所谓的 5%规则：该规则对于多数组合优化问题，若令 5%的修复过的染色体替代原染色体，则带有修复程序的遗传算法可取得最好的效果[322]。Michalewicz 等则认为对有非线性约束的优化问题，15%的替代律为

最好[291]。

3. 改进遗传算子策略

解决可行性问题的一个合理办法是设计针对问题的表达方式以及专门的遗传算子来维持染色体的可行性。Michalewicz 等指出这种方法通常比基于惩罚的遗传算法更可靠。许多领域中的实际工作者采用专门的问题表达方式和遗传算子构造了非常成功的遗传算法，这已是一个十分普遍的趋势[291]。但是，该方法的遗传搜索受到了可行域的限制。

4. 惩罚策略

上面三种策略的共同优点是都不会产生不可行解，缺点则是无法考虑可行域外的点。对于约束严的问题，不可行解在种群中的比例很大。这样，将搜索限制在可行域内就很难找到可行解。Glover 和 Greenberg 建议的约束管理技术允许在搜索空间里的不可行域中进行搜索，这比将搜索限制在可行域内的方法能更快地获得最优解或获得更好的最终解[170]。惩罚策略就是这类在遗传搜索中考虑不可行解的技术。

2.2.2 惩罚函数

惩罚技术大概是用遗传算法解约束优化问题中最常用的技术。本质上它是通过惩罚不可行解将约束问题转化为无约束问题。在遗传算法中，惩罚技术用来在每代的种群中保持部分不可行解，使遗传搜索可以从可行域和不可行域两边来达到最优解。

一般，解空间包含两部分：可行域和不可行域。我们不能对这些子空间作任何假设，特别是当它们是非凸或非连通时，如图 2.2 所示。控制非可行的染色体远不是一件容易事，从图 2.2 中可知，不可行解 b 与最优解 a 的距离比不可行解 d 和可行解 c 都近。我们希望给 b 较小惩罚，即使它比 d 离可行域更远。可以相信尽管 b 是不可行解，但它比 c 含有更多的有关最优点的信息。然而，我们对最优点没有任何先验知识，所以一般很难判断哪一点更好。

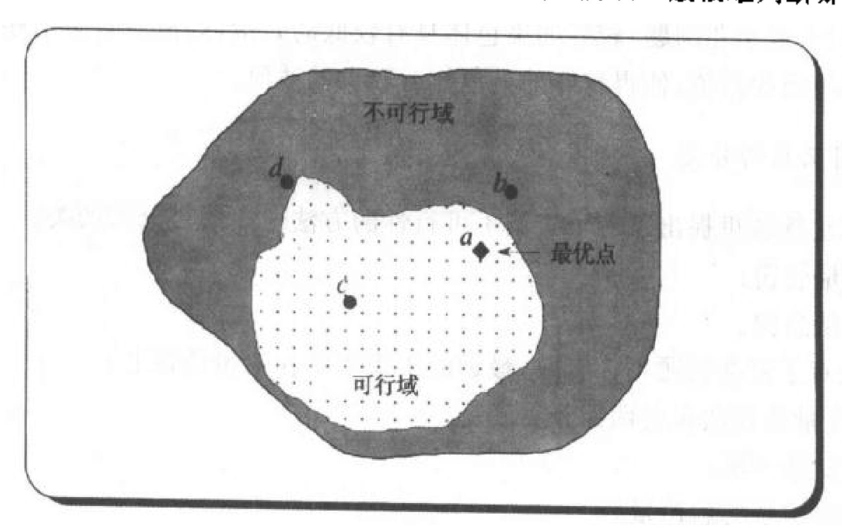


图 2.2 解空间：可行域与不可行域

惩罚策略的主要问题是如何设计一个惩罚函数 $p(\mathbf{x})$，从而能有效地引导遗传搜索达到解空间的最好区域。不可行染色体和解空间可行部分的关系在惩罚不可行染色体中起

了关键作用:不可行染色体的惩罚值相应于某种测度下的不可行性的"测量"。设计惩罚函数没有一般规则,这仍要依赖于待解的问题。

1. 惩罚项的评估函数

惩罚技术通过惩罚不可行解将约束问题转化为无约束问题。构造带有惩罚项的评估函数的方法一般有两种。一种采用加法形式:

$$eval(\mathbf{x}) = f(\mathbf{x}) + p(\mathbf{x}) \tag{2.5}$$

其中,$\mathbf{x}$ 代表染色体;$f(\mathbf{x})$是问题的目标函数;$p(\mathbf{x})$是惩罚项。对于极大化问题,取

$$\begin{cases} p(\mathbf{x}) = 0, & \text{若 } \mathbf{x} \text{ 可行} \\ p(\mathbf{x}) < 0, & \text{其他} \end{cases} \tag{2.6}$$

令$|p(\mathbf{x})|_{max}$和$|f(\mathbf{x})|_{min}$分别为现行种群中的$|p(\mathbf{x})|$的最大值和$|f(\mathbf{x})|$的最小值。并要求

$$|p(\mathbf{x})|_{max} \leqslant |f(\mathbf{x})|_{min} \tag{2.7}$$

以避免出现负的适值。对于极小化问题,则取

$$\begin{cases} p(\mathbf{x}) = 0, & \text{若 } \mathbf{x} \text{ 可行} \\ p(\mathbf{x}) > 0, & \text{其他} \end{cases} \tag{2.8}$$

另一种方法是采用乘法形式:

$$eval(\mathbf{x}) = f(\mathbf{x})\, p(\mathbf{x}) \tag{2.9}$$

这时,极大化问题可取为

$$\begin{cases} p(\mathbf{x}) = 1, & \text{若 } \mathbf{x} \text{ 可行} \\ 0 \leqslant p(\mathbf{x}) < 1, & \text{其他} \end{cases} \tag{2.10}$$

对极小化问题则取

$$\begin{cases} p(\mathbf{x}) = 1, & \text{若 } \mathbf{x} \text{ 可行} \\ p(\mathbf{x}) > 1, & \text{其他} \end{cases} \tag{2.11}$$

注意,对于极小化问题,较好的染色体具有较低的 $eval(\mathbf{x})$值。对于某些选择方法,需要将目标值转换为适值,使得较好的染色体有较大的适值。

2. 惩罚函数的分类

遗传算法领域里提出了若干控制不可行性的方法。一般可分为两类:

(1) 定量惩罚;

(2) 变量惩罚。

定量惩罚法对于复杂问题不太有效,最近的工作集中在变量惩罚上。

一般,变量惩罚法包含两部分:

(1) 可变惩罚率;

(2) 违反约束的惩罚量。

可变惩罚率可按以下因素调节:

(1) 约束违反程度;

(2) 遗传算法的迭代次数。

Michalewicz 指出,前者随约束违反变得严重而增加惩罚压力,属于静态惩罚;后者随

进化过程的进展而增加惩罚压力，属于动态惩罚[289]。

基本上，惩罚是距可行域的距离的函数。这个距离可按以下三种方式测量：

(1) 单一不可行解的绝对距离的函数；

(2) 现行种群中所有不可行解的相对距离的函数；

(3) 自适应惩罚项的函数。

多数算法采用第一种方法。对于约束严的问题，每代中不可行解与可行解的比相对较高。这时，第二和第三种方法有可能在保留信息和增加对不可行性的压力之间寻求折衷。

惩罚法可进一步分为

(1) 问题依赖的；

(2) 问题独立的。

多数惩罚技术属于前者。

惩罚法还可分为

(1) 带参数的；

(2) 不带参数的。

多数惩罚技术属于带参数的。参数惩罚函数似乎都是问题依赖的。

按照惩罚函数的分类，我们介绍几种求解非线性规划问题的遗传算法的构造惩罚函数的方法。

(1) Homaifar，Qi 和 Lai 方法

Homaifar 等考虑如下非线性规划问题[222]：

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \geqslant 0;\ i = 1,2,\cdots,m \end{aligned}$$

取加法形式的惩罚函数

$$eval(\mathbf{x}) = f(\mathbf{x}) + p(\mathbf{x})$$

惩罚函数由两部分构成：①可变乘法因子和②违反约束惩罚。其表达式如下：

$$p(\mathbf{x}) = \begin{cases} 0, & \text{若 } \mathbf{x} \text{ 可行} \\ \sum_{i=1}^{m} r_i\, g_i(\mathbf{x}), & \text{其他} \end{cases} \tag{2.12}$$

其中 r_i 是约束 i 的可变惩罚系数。对于每个约束，又可分为几个违反级，按违反的级，r_i 相应变化。然而，对每个约束确定违反级和适当的 r_i 的值也不是一件容易的事，这要依赖于要解的问题。

Michalewicz 最近指出，解的质量严重地依赖于这些惩罚系数的值[288]。当惩罚系数不适当时，算法可能收敛于不可行解；另一方面，惩罚系数过大时该方法等价于拒绝策略。

(2) Joines 和 Houck 方法

Joines 和 Houck 考虑如下非线性规划问题[237]：

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \geqslant 0;\ i = 1,2,\cdots,m_1 \\ & h_i(\mathbf{x}) = 0;\ i = m_1 + 1,2,\cdots,m\ (= m_1 + m_2) \end{aligned}$$

取加法形式的评估函数：

$$eval(\mathbf{x}) = f(\mathbf{x}) + p(t,\mathbf{x})$$

惩罚函数也由两部分构成:①可变乘法因子和②违反约束的惩罚。其表示式如下:

$$p(t,\mathbf{x}) = \rho_t^{\alpha}\sum_{i=1}^{m} d_i^{\beta}(\mathbf{x}) \tag{2.13}$$

这里,t 是遗传算法的迭代次数,α 和 β 是调节惩罚值大小的参数。单个约束 $d_i(\mathbf{x})$的惩罚项和惩罚因子按下式计算:

$$d_i(\mathbf{x})=\begin{cases}0, & \text{若 } \mathbf{x} \text{ 可行}\\ |g_i(\mathbf{x})|, & \text{否则 } 1\leqslant i\leqslant m_1\\ |h_i(\mathbf{x})|, & \text{否则 } m_1+1\leqslant i\leqslant m\end{cases} \tag{2.14}$$

$$\rho_t = C\times t \tag{2.15}$$

其中 C 是常数。通过 ρ_t 对不可行染色体的惩罚随进化过程进展而增加。与 Homaifar, Qi 和 Lai 的方法相比,两方法的差别是 Joine 和 Houck 的方法中可变惩罚因子随迭代次数而变,而在 Homaifar, Qi 和 Lai 的方法中则随违反级变。

Joines 和 Houck 的实验结果表明,解的质量对三个参数的值很灵敏。如何确定可变参数项是依赖问题的。使可变部分具有适合给定问题的适当的动态性质是十分必要的,因为在遗传算法较后的代中,该方法将给不可行染色体以死亡惩罚。按 Michalewicz 的经验,该方法会因惩罚过大而过早收敛[289]。

(3) Michalewicz 和 Attia 方法

Michalewicz 和 Attia 考虑如下非线性规划问题[289]:

$$\begin{aligned}\min\quad & f(\mathbf{x})\\ \text{s.t.}\quad & g_i(\mathbf{x}) > 0;\ i=1,2,\cdots,m_1\\ & h_i(\mathbf{x}) = 0;\ i=m_1+1,\cdots,m(=m_1+m_2)\end{aligned}$$

取加法形式的评估函数:

$$eval(\mathbf{x}) = f(\mathbf{x}) + p(\tau,\mathbf{x})$$

惩罚函数也由两部分构成:①可变惩罚因子和②违反约束的惩罚,其表示式如下:

$$p(\tau,\mathbf{x}) = \frac{1}{2\tau}\sum_{i\in A} d_i^2(\mathbf{x}) \tag{2.16}$$

这里,A 是起作用的约束集,它由所有等式约束和不能满足的不等式约束构成。约束$g_i(\mathbf{x})$在点 $\mathbf{x}$ 被违反或不满足,当且仅当 $g_i(\mathbf{x}) > \delta$ $(i=m_1+1,\cdots,m)$,其中 δ 是决定该约束是否起作用的参数。τ 是可变惩罚因子,称为温度。对单一约束 $d_i(\mathbf{x})$的惩罚按下式给出:

$$d_i(\mathbf{x})=\begin{cases}\max\{0,\ g_i(\mathbf{x})\}, & \text{若 } 1\leqslant i\leqslant m_1\\ |h_i(\mathbf{x})|, & \text{若 } m_1+1\leqslant i\leqslant m\end{cases} \tag{2.17}$$

他们用以上技术构造了一个系统,称为 Genocop II [290]。

注意,Genocop II 在机理上像遗传算法但更像模拟退火。可变因子 τ,从初始温度 τ_0 开始,终止在冻结温度 τ_f,在主循环中按给定的冷却程序逐步降温。主循环中嵌入了一个找改进点的程序 Genocop I。在 Genocop I 的每次执行中 τ 都保持不变。这个方法对参数值十分灵敏。存在的问题是对具体问题应如何设定参数。

(4) Smith, Tate 和 Coit 方法

自适应惩罚函数是由 Smith 和 Tate 首先提出的[382],该惩罚函数能根据获得的最好解的适值动态地标定惩罚的量级。随着较好的可行解和不可行解被找到,加在某给定不

可行解上的惩罚可以改变。Coit 和 Smith 进一步扩展了以上工作[82],并提出近可行性阈(Near-Feasibility Threshold)的概念，简称(NFT)。外部惩罚函数为给定解到可行域的“距离”的非减函数。NFT 是一个与可行域距离有关的阈值,距离小于 NFT 后认为搜索“变热”。该惩罚函数将鼓励在可行域和 NFT 的邻域内的搜索,而限制在阈值界定的邻域外的搜索。

他们考虑如下非线性规划问题：

$$\max \quad f(\mathbf{x})$$
$$\text{s.t.} \quad g_i(\mathbf{x}) \leqslant b_i;\ i = 1, 2, \cdots, m$$

取加法形式的评估函数：

$$eval(\mathbf{x}) = f(\mathbf{x}) + p(\mathbf{x})$$

惩罚函数由两部分构成:①违反约束的相对惩罚系数和②自适应惩罚项,其表示式如下：

$$p(\mathbf{x}) = -\sum_{i=1}^{m}\left(\frac{\Delta b_i(\mathbf{x})}{\Delta b_i^{nef}}\right)^{\alpha}(f_{all}^{*} - f_{feas}^{*}) \tag{2.18}$$

其中,α 是用来调节惩罚严厉性的参数;$\Delta b_i(\mathbf{x})$为约束 i 的违反量;Δb_i^{nef} 是约束 i 的近可行性阈,如何确定适当的 nef 取决于具体问题;f_{feas}^{*}是已经找到的最好可行解的目标函数值,而 f_{all}^{*}是已获得的未受惩罚的最好解的目标函数值。

注意,自适应项($f_{all}^{*} - f_{feas}^{*}$)可能带来两个问题:①零惩罚和②过惩罚。当 $f_{all}^{*} = f_{feas}^{*}$ 时,即使有不可行解,该方法对所有不可行解也给予零惩罚;而在进化初始阶段出现某个不可行解的目标值 f_{all}^{*}很大时,就会给所有不可行解加上过大的惩罚。

(5) Yokota, Gen, Ida 和 Taguchi 方法

Yokota, Gen, Ida 和 Taguchi 考虑了与 Smith, Tate 和 Coit 研究过的同样的非线性规划问题,但取的是乘法形式的评估函数[151]:

$$eval(\mathbf{x}) = f(\mathbf{x})\ p(\mathbf{x})$$

惩罚函数构成如下：

$$p(\mathbf{x}) = 1 - \frac{1}{m}\sum_{i=1}^{m}\left(\frac{\Delta b_i(\mathbf{x})}{b_i}\right)^{\alpha} \tag{2.19}$$

$$\Delta b_i(\mathbf{x}) = \max\ \{0,\ g_i(\mathbf{x}) - b_i\} \tag{2.20}$$

其中,$\Delta b_i(\mathbf{x})$是约束 i 的违反量。该惩罚函数可以看成是 Smith, Tate 和 Coit 方法的特例,这里约束 i 的近可行性阈(NFT)设为 $\Delta b_i^{nef} = b_i$,所以该方法的惩罚比 Smith 等的方法相对温和一些。特别应注意,该惩罚函数是不含参数的,且不依赖于问题。

(6) Gen 和 Cheng 方法

为了给不可行解更严厉的惩罚,Gen 和 Cheng 进一步改进了以上方法[151]:令 $\mathbf{x}$ 是当前种群 $P(t)$中的一个染色体。惩罚函数构造如下：

$$p(\mathbf{x}) = 1 - \frac{1}{m}\sum_{i=1}^{m}\left(\frac{\Delta b_i(\mathbf{x})}{\Delta b_i^{max}}\right)^{\alpha} \tag{2.21}$$

这里

$$\Delta b_i(\mathbf{x}) = \max\ \{0,\ g_i(\mathbf{x}) - b_i\} \tag{2.22}$$

$$\Delta b_i^{max} = \max\ \{\epsilon,\ \Delta b_i(\mathbf{x});\ x \in P(t)\ \} \tag{2.23}$$

其中,$\Delta b_i(\mathbf{x})$是约束 i 在 $\mathbf{x}$ 的违反量;Δb_i^{max} 是当前种群中约束 i 的最大违反量;ϵ 为避免

除零的小正数。对严约束优化问题，每代中不可行解在种群中的比例较大。该惩罚方法可以逐代自动调节惩罚比，以维持信息保留和不可行惩罚压力的平衡，从而避免过惩罚。

2.2.3 遗传运算

对于多数遗传算法在约束优化中的应用问题，常用实数编码技术来表达给定问题的解。在实数编码中，每个染色体编码为一个和解向量维数相同的实向量。这种编码方法称为浮点表达[287]，实数表达[101]或连续表达[302]。例如，对于优化问题 (2.1)～(2.4)，实向量 $\mathbf{x}=[x_1,x_2,\cdots,x_n]$就用作表达解的染色体。

近年来，已提出多种采用以上编码方法的遗传算法，其遗传运算方式粗略可分为以下几类：

(1) 传统运算；

(2) 算术运算；

(3) 基于方向的运算。

将二进制表达的运算扩展到实数表达即为传统运算。算术运算则是借用凸集理论中向量线性组合的概念。基于方向的运算是将近似梯度(次梯度)或负的梯度方向引入遗传算法得来的。

1. 传统运算

(1) 简单交叉

一般的交叉运算可模仿二进制表达的交叉[101]。最基本的是单点交叉，令双亲为 $\mathbf{x}=[x_1,x_2,\cdots,x_n]$和 $\mathbf{y}=[y_1,y_2,\cdots,y_n]$。在随机的第 k 位交叉，生成的后代为

$$\mathbf{x}'=[x_1,x_2,\cdots,x_k,y_{k+1},y_{k+2},\cdots,y_n]$$

$$\mathbf{y}'=[y_1,y_2,\cdots,y_k,x_{k+1},x_{k+2},\cdots,x_n]$$

进一步，可将二进制表达的两点、多点和均匀交叉扩展到实数表达中，详见 Spears 和 De Jong 的文献[385]和 Syswerda 的文献[391]。

(2) 随机交叉

本质上，该交叉运算是在由双亲界定的超矩形中随机地产生后代。其基本方法由 Radcliffe 给出[343]，称为浮点交叉。对于每个基因，它在双亲的两个基因值之间按均匀分布随机地取一个值作为后代的相应基因的值。Eshelman 和 Schaffer 扩展了 Radcliffe 的工作[121]，他们引入较大的方差，即在包含双亲的两点间均匀地选取后代的基因值，称之为盲目交叉。

(3) 变异

一般表达的变异运算和二进制表达的大不相同，作为实数基因可在一定范围内变异。最基本的一种是均匀变异，即简单地在指定范围内随机选一个实数替代原基因。令要变异的染色体为 $\mathbf{x}=[x_1,x_2,\cdots,x_n]$。首先选一随机整数 $k\in[1,n]$，然后产生后代 $\mathbf{x}'=[x_1,\cdots,x'_k,\cdots,x_n]$，其中，$x'_k$ 是$[x_k^L,x_k^U]$中均匀分布的一个随机值。x_k^L 和 x_k^U 通常可取为变量 x_k 的上下界，一般可由约束域确定，也可以根据约束集(不等式)动态计算[234]。

基因 x'_k 也可以等概率地用 x_k^L 或 x_k^U 替代。这种变型称为边界变异[292]。也可用$[x_{k-1},x_{k+1}]$来替代上下界，这种变形称为简易变异[234]。

2. 算术运算

(1) 交叉

这种运算的基本概念源于凸集理论[26]。一般,两向量 $\mathbf{x}_1$ 和 $\mathbf{x}_2$ 的加权平均计算如下:

$$\lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2 \tag{2.24}$$

若乘子限制为

$$\lambda_1 + \lambda_2 = 1, \qquad \lambda_1 > 0, \lambda_2 > 0$$

则加权形式(2.24) 称为凸组合。若去掉非负限制,则称为仿射组合。如果仅要求乘子属于实数空间 E_1, 则称为线性组合。

类似地,算术运算被定义为两个向量(染色体)的如下组合:

$$\mathbf{x}_1' = \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2$$

$$\mathbf{x}_2' = \lambda_1 \mathbf{x}_2 + \lambda_2 \mathbf{x}_1$$

按对乘子限制的不同,可获得三种交叉,相应地称为凸交叉、仿射交叉和线性交叉。

凸交叉最为常用[287]。当取 $\lambda_1=\lambda_2=0.5$, 便得到它的特例,Davis 称为平均交叉[101],Schwefel 则称为中间交叉[372]。

仿射交叉首先是由 Wright 提出的[421],对一个特例,乘子取为

$$\lambda_1 = 1.5 \text{ 和 } \lambda_2 = -0.5$$

Mühlenbein 和 Schlierkamp-Voosen 提出了另一种仿射交叉, 称为扩展中间交叉[302], 其中一个乘子取为区间$[-d,1+d]$中的一个随机实数。

线性交叉(采用不同于凸交叉和仿射交叉的严格定义)是由 Cheng 和 Gen 首先提出的[67]。他们限制乘子为

$$\lambda_1 + \lambda_2 \leqslant 2, \qquad \lambda_1 > 0, \lambda_2 > 0$$

下面介绍算术运算的几何解释。对于双亲 $\mathbf{x}_1$ 和 $\mathbf{x}_2$, 其所有凸组合的集合称为凸壳(convex hull)。类似地,称所有仿射组合的集合为仿射壳,所有线性组合的集合称为线性壳。图 2.3 显示了二维空间的简单情况。凸交叉产生的后代位于实线段,仿射交叉的后代位于虚线段,而线性交叉的后代分布在整条直线上。

读者可能会想到仿射交叉和线性交叉会产生不可行的后代,但这并不是一个严重问题,在遗传搜索中使用有效的罚函数,可以允许进化过程中包含不可行解,关键是如何确定适当的乘子值。一般,对于一个具体问题运用仿射或线性交叉时,应根据约束域确定乘子的上下界,使遗传搜索控制在合理的范围内。

(2) 动态变异

动态变异运算又称为非均匀变异,是由 Janilow 和 Michalewicz 提出的[234]。它是为提高精度,增加细调能力而设计的。对于父亲 $\mathbf{x}$, 若元素 x_k 被选出作变异,则后代为 $\mathbf{x}' = [x_1,\cdots,x_k',\cdots,x_n]$, 其中 x_k' 是按如下两种可能选得的:

$$x_k' = x_k + \Delta(t, x_k^U - x_k)$$

或

$$x_k' = x_k - \Delta(t, x_k - x_k^L)$$

函数 $\Delta(t,y)$返回$[0,y]$中的一个值,使得 $\Delta(t,y)$随 t 增加而趋于 0 (t 是代数)。函数的这个性质使得初始迭代时,搜索均匀分布在整个空间,而到后期则分布在局部范围内。

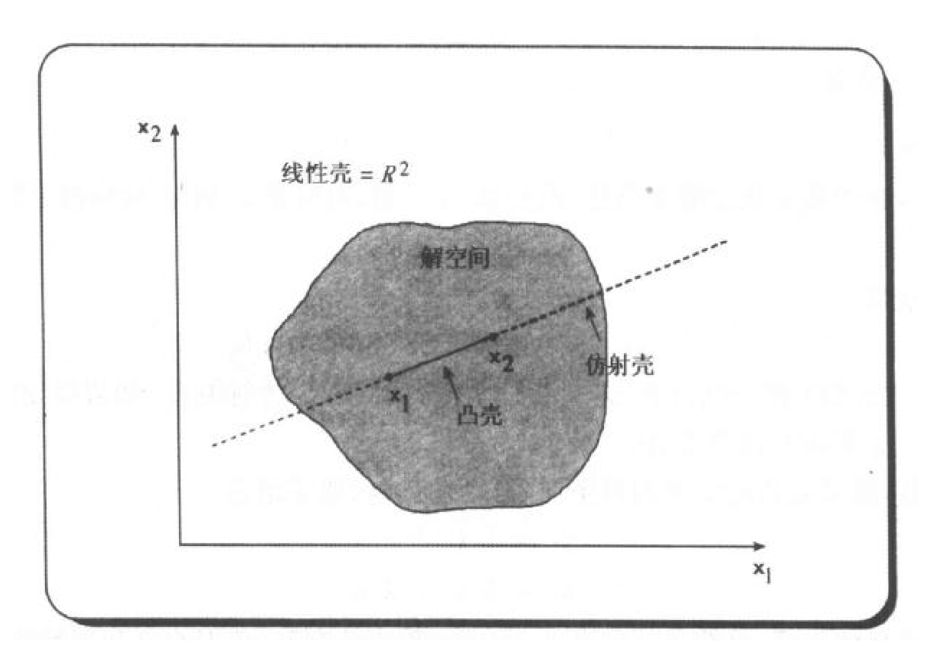


图 2.3 凸壳,仿射壳和线性壳的说明

$\Delta(t,y)$给出如下:

$$\Delta(t,y) = y \cdot r \cdot \left(1 - \frac{t}{T}\right)^b$$

其中,r 是[0,1]间的随机数;T 是最大代数;b 是确定不均匀度的参数。该运算可能产生不可行的后代,这时可以减小随机数 r。

3. 基于方向的运算

以上讨论的几种遗传运算都不能保证后代比双亲更好。为获得好于双亲的后代,基于方向的运算,将问题的知识引入遗传算法。

(1) 交叉

基于方向的交叉采用目标函数值来确定遗传搜索的方向[292]。该运算按以下公式由双亲 $\mathbf{x}_1$ 和 $\mathbf{x}_2$ 产生一个后代 $\mathbf{x}'$:

$$\mathbf{x}' = r \cdot (\mathbf{x}_2 - \mathbf{x}_1) + \mathbf{x}_2$$

式中,r 是 0 和 1 之间的随机数。假设双亲中 $\mathbf{x}_2$ 不比 $\mathbf{x}_1$ 坏,即对极大问题,$f(\mathbf{x}_2) \geqslant f(\mathbf{x}_1)$,或对极小问题,$f(\mathbf{x}_2) \leqslant f(\mathbf{x}_1)$。

(2) 变异

基于方向的变异由 Gen 和 Liu 提出[163,164]。连续可微函数 f 的 Taylor 展开为

$$f(\mathbf{x} + \Delta\mathbf{x}) \cong f(\mathbf{x}) + (\nabla f(\mathbf{x} + \theta\Delta\mathbf{x}))^T \Delta\mathbf{x}, \qquad \mathbf{x} \in R^n$$

式中,$0 \leqslant \theta \leqslant 1$;$\nabla f(\mathbf{x})$为函数 f 在点 $\mathbf{x}$ 的梯度;$\Delta\mathbf{x}$ 是 R^n 中的摄动。对于极大问题选梯度方向变异较好,而对极小问题选负梯度方向作变异方向较好。由于遗传算法不像传统方法那样要求问题具有很好的数学性质,因此,可以按以下公式计算近似梯度的第 i 个分量:

$$\frac{f(x_1,\cdots,x_i + \Delta x_i,\cdots,x_n) - f(x_1,\cdots,x_i,\cdots,x_n)}{\Delta x_i}$$

其中,Δx_i 是一个小的实数。令 $\mathbf{d}$ 是由上式计算的近似梯度,变异后的后代即为

$$\mathbf{x}' = \mathbf{x} + r \cdot \mathbf{d}$$

其中 r 随机的非负实数,但选择时应使后代是可行解。

为避免染色体都挤到一个角落里,有时也可随机地选一个自由方向。因为,当染色体靠近边界时,按上述方法给出的变异方向可能都指向边界,这时就会发生拥挤。这种情况下,就应用随机方向替代给定的方向。

2.2.4 计算例子

例 2.1 考虑如下问题:

$$\begin{aligned} \min \quad & f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2 \\ \text{s.t.} \quad & g_1(\mathbf{x}) = x_1 - 2x_2 + 1 = 0 \\ & g_2(\mathbf{x}) = x_1^2 / 4 - x_2^2 + 1 \geqslant 0 \end{aligned}$$

该问题源于 Bracken 和 McCormick 的书[47]。由 Homaifar, Qi 和 Lai 用遗传算法求解[222]。惩罚函数为 $P(\mathbf{x}) = r_1 g_1(\mathbf{x}) + r_2 g_2(\mathbf{x})$,其中,$r_1$ 和 r_2 是惩罚因子(详见文献[222])。遗传算法的参数选择为:染色体长度 = 19,pop_size = 400,p_c= 0.85, p_m= 0.02. 100 代后得到的解见表 2.1,精度为 0.0284. 验证表明遗传算法获得的解满足两个约束,其中,GRG 代表广义简约梯度法[146]。

表 2.1 数值实验的结果

项目	参考解	遗传算法的解	GRG 的解
$f(\mathbf{x})$	1.393	1.4339	1.3934
x_1	0.823	0.8080	0.8229
x_2	0.911	0.88544	0.9115
$g_1(\mathbf{x})$	1.0×10^{-3}	3.7×10^{-2}	1.00×10^{-4}
$g_2(\mathbf{x})$	7.46×10^{-4}	0.052	-5.18×10^{-5}

例 2.2 考虑如下问题:

$$\begin{aligned} \min \quad & f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \\ \text{s.t.} \quad & 0 \leqslant 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5 \leqslant 92 \\ & 90 \leqslant 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leqslant 110 \\ & 20 \leqslant 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leqslant 25 \\ & 78 \leqslant x_1 \leqslant 102 \\ & 33 \leqslant x_2 \leqslant 45 \\ & 27 \leqslant x_3 \leqslant 45 \\ & 27 \leqslant x_4 \leqslant 45 \\ & 27 \leqslant x_5 \leqslant 45 \end{aligned}$$

这是 Himmelblau 提供的一个有趣的非线性优化问题[215]。它有 5 个自变量,6 个非线性约束和 10 个上下界约束。注意,x_2 和 x_4 未显式地包含在目标函数里。Homaifar, Qi 和 Lai 用遗传算法求解[222],参数选择如下:染色体长度= 19,pop_size = 400, p_c= 0.8,p_m= 0.088。结果如表 2.2 所示,它表明基于局部参考的解稍微好于全局参考解。两种解都满足约束条件。

表 2.2 数值实验的结果

项目	参考解	遗传算法基于全局参考的解	遗传算法基于局部参考的解	GRG 的解
$f(\mathbf{x})$	−30665.5	−30175.804	−30182.269	−30373.950
x_1	78.00	80.61	81.49	78.62
x_2	33.00	34.21	34.09	33.44
x_3	29.995	31.34	31.24	31.07
x_4	45.00	42.05	42.20	44.18
x_5	36.776	34.85	34.37	35.22

例 2.3 考虑如下问题：

$$\min \quad f(\mathbf{x}) = \sum_{j=1}^{10} x_j\left(c_j + \ln \frac{x_j}{x_1 + \cdots + x_{10}}\right)$$

$$\begin{aligned} \text{s.t.} \quad & x_1 + 2x_2 + 2x_3 + x_6 + x_{10} = 2 \\ & x_4 + 2x_5 + x_6 + x_7 = 1 \\ & x_3 + x_7 + x_8 + 2x_9 + x_{10} = 1 \\ & x_i \geqslant 0.000001 (i = 1, \cdots, 10) \end{aligned}$$

式中

$c_1 = -6.089, c_2 = -17.164, c_3 = -34.054, c_4 = -5.914, c_5 = -24.721$

$c_6 = -14.986, c_7 = -24.100, c_8 = -10.708, c_9 = -26.662, c_{10} = -22.179$

这个问题是 Hock 和 Schittlowski 给出的[219]，Michalewicz，Logan 和 Swaminathan 则用遗传算法进行了求解[292]。已知最好解为

$$\mathbf{x}^* = [0.01773548, 0.08200180, 0.8825646, 0.0007233256, 0.4907851, 0.0004335469, 0.01727298, 0.007765639, 0.01984929, 0.05269826]$$

和

$$f(\mathbf{x}^*) = -47.707579$$

Michalewicz 等经 10 次运行发现了一个更好的解：

$$\mathbf{x}^* = [0.04034785, 0.15386976, 0.77497089, 0.00167479, 0.48468539, 0.00068965, 0.02826479, 0.01849179, 0.03849563, 0.10128126]$$

其目标函数值为−47.760765。一次运行 500 代的计算需要 11s CPU 时间。

例 2.4 考虑如下问题：

$$\min \quad f(x, \mathbf{y}) = 6.5x - 0.5x^2 - y_1 - 2y_2 - 3y_3 - 2y_4 - y_5$$

$$\begin{aligned} \text{s.t.} \quad & x + 2y_1 + 8y_2 + y_3 + 3y_4 + 5y_5 \leqslant 16 \\ & -8x - 4y_1 - 2y_2 + 2y_3 + 4y_4 - y_5 \leqslant -1 \\ & 2x + 0.5y_1 + 0.2y_2 - 3y_3 - y_4 - 4y_5 \leqslant 24 \\ & 0.2x + 2y_1 + 0.1y_2 - 4y_3 + 2y_4 + 2y_5 \leqslant 12 \\ & -0.1x - 0.5y_1 + 2y_2 + 5y_3 - 5y_4 + 3y_5 \leqslant 3 \\ & y_3 \leqslant 1, \quad y_4 \leqslant 1, \quad y_5 \leqslant 2 \\ & x \geqslant 0, \quad y_i \geqslant 0, \quad 1 \leqslant i \leqslant 5 \end{aligned}$$

该问题源于 Floudas 和 Pardalos 的工作[127]。Michalewicz，Logan 和 Swaminathan 用遗传算法求解[292]，全局解为$[x, y^*] = [0,6,0,1,1,0]$，和$f(x, y^*) = -11.005$. Michalewicz 等 10 次运行求得的解和上述最优解十分接近：

$$[0.000000, 5.976089, 0.005978, 0.999999, 1.000000, 0.000000]$$

其函数目标值为 -10.988042。一次运行 1000 代占用 29s CPU 时间。

例 2.5 考虑如下问题：

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x}) + f_3(\mathbf{x}) \\
\text{s.t.} \quad & x_1 / \sqrt{3} - x_2 \geqslant 0 \\
& -x_1 - \sqrt{3} x_2 + 6 \geqslant 0 \\
& 0 \leqslant x_1 \leqslant 6,\ x_2 \geqslant 0
\end{aligned}$$

式中

$$f_1(\mathbf{x}) = x_2 + 10^{-5}(x_2 - x_1)^2 - 1.0, \quad 0 \leqslant x_1 \leqslant 2$$

$$f_2(\mathbf{x}) = \frac{1}{27\sqrt{3}}((x_1 - 3)^2 - 9)x_2^3, \quad 2 \leqslant x_1 \leqslant 4$$

$$f_3(\mathbf{x}) = \frac{1}{3}(x_1 - 2)^3 + x_2 - \frac{11}{3}, \qquad 4 \leqslant x_1 \leqslant 6$$

该问题是 Hock 和 Schittlowski 提出的[219]，Michalewicz，Logan 和 Swaminathan 用遗传算法进行了求解[292]。该问题由三个子问题构成，函数 f 有三个全局解：

$$\mathbf{x}_1^* = [0,0], \quad \mathbf{x}_2^* = [3, \sqrt{3}], \quad \mathbf{x}_3^* = [4,0]$$

三个最优值都是 $f(\mathbf{x}_i^*) = -1$，$i = 1,2,3$。Michalewicz 等分别做了三个实验，其结果为：实验 1 的全局解为 $\mathbf{x}_1^* = [0,0]$，实验 2 的是 $\mathbf{x}_2^* = [3, \sqrt{3}]$，实验 3 的是 $\mathbf{x}_3^* = [4, 0]$。三个实验寻找全局最优的时间都是一次运行 500 代需要 9s CPU 时间。

2.3 随机优化

数学规划在实际应用中的一个共同问题是难于确定适当的模型参数值，甚至模型的解已经求出并使用后，模型参数的真实值还是不清楚。这有时可以归结于调查研究的不充分。然而，这些参数的值通常受一些不可预测的随机事件的影响而发生波动，即模型的部分或全部参数是随机变量。因此需要有一种方法来描述这类问题，使得直接处理不确定性环境下的优化问题成为可能。其方法之一就是随机规划[214,239]。

2.3.1 数学模型

下面的讨论集中在随机规划上。若 $\mathbf{x}$ 是实向量，ξ 是随机向量，函数 $f(\mathbf{x},\xi)$对于给定的 $\mathbf{x}$ 是随机变量。由于极大化随机变量没有意义，因此，人们自然想到用其期望值 $E[f(\mathbf{x},\xi)]$来代替。于是，随机规划问题可一般描述如下：

$$\max \quad E[f(\mathbf{x},\xi)] \tag{2.25}$$

$$\text{s.t.} \quad E[g_i(\mathbf{x},\xi)] \leqslant 0; \quad i = 1,2,\cdots,m_1 \tag{2.26}$$

$$E[h_i(\mathbf{x},\xi)] = 0; \quad i = m_1 + 1,\cdots, m(= m_1 + m_2) \tag{2.27}$$

其中，E 表示取期望值；$\mathbf{x} = [x_1, x_2, \cdots, x_n]$是 n 维实向量；$\xi = [\xi_1, \xi_2, \cdots, \xi_l]$是 l 维随机向

量；$f, g_i(i=1,2,\cdots,m_1), h_i(i=m_1+1,\cdots,m)$是定义在 R^{n+l}上的实值函数。

若随机向量 ξ 的分布函数和密度函数分别为 $\Phi(\xi)$和 $\phi(\xi)$，于是有：

$$E[f(\mathbf{x},\xi)] = \int_{R^m} f(\mathbf{x},\xi) d\Phi(\xi) = \int_{R^m} f(\mathbf{x},\xi)\phi(\xi)\, d\xi \tag{2.28}$$

$$E[g_i(\mathbf{x},\xi)] = \int_{R^m} g_j(\mathbf{x},\xi) d\Phi(\xi) = \int_{R^m} g_i(\mathbf{x},\xi)\phi(\xi)\, d\xi;\ i = 1,2,\cdots,m_1 \tag{2.29}$$

$$E[h_i(\mathbf{x},\xi)] = \int_{R^m} h_j(\mathbf{x},\xi) d\Phi(\xi) = \int_{R^m} h_i(\mathbf{x},\xi)\phi(\xi)\, d\xi;\ i = m_1+1,\cdots,m \tag{2.30}$$

解这个问题需要花费大量的精力进行多元积分，而且积分域很难确定，特别是维数较高和/或约束复杂时。

2.3.2 Monte Carlo 仿真

Monte Carlo 仿真广泛地用来求解不可用解析方法解决的统计学中的确定型问题。Monte Carlo 仿真是应用随机数，即独立同分布的随机变量来近似问题解的过程[262]。虽然该技术称之为仿真，其实它不包含模拟实际过程的意义。它的一个标准的应用是计算积分。

假设我们要作如下积分：

$$I = \int_a^b g(x)\, dx$$

其中，$g(x)$是不可解析积分的实值函数。为说明如何用 Monte Carlo 仿真作这个确定型积分，令 Y 为随机变量$(b-a)g(X)$，其中 X 是连续型随机变量，在 $[a, b]$上均匀分布，记为 $U(a, b)$。于是，Y 的期望值为

$$\begin{aligned} E[Y] &= E[(b-a)g(X)] \\ &= (b-a)E[g(X)] \\ &= (b-a)\int_a^b g(x)\, f_X(x)\, dx \\ &= \int_a^b g(x)\, dx \\ &= I \end{aligned}$$

其中，$f_X(x) = 1/(b-a)$是 $U(a, b)$分布的随机变量的概率密度函数。于是，积分问题就转换成了计算期望值 $E(Y)$的问题。实际上可用以下简化的方法来求这个期望值：

$$\overline{Y}(n) = \frac{1}{n}\sum_{i=1}^{n} Y_i = \frac{(b-a)}{n}\sum_{i=1}^{n} g(X_i)$$

其中 $X_1, X_2, \cdots, X_n$ 是独立并同为 $U(a,b)$分布的随机变量。可以证明 $\overline{Y}(n)$是积分 I 的一个无偏估计，即 $E[\overline{Y}(n)]=I$，且 $\mathrm{Var}[\overline{Y}(n)]=\mathrm{Var}[Y]/n$。假设 $\mathrm{Var}[Y]$是有限的，那么对于充分大的 n，$\overline{Y}(n)$将以概率 1 任意接近积分 I。

实际上，不可能用 Monte Carlo 仿真来作这类单一积分，因为这类积分已有很多有效方法。下面说明如何作随机积分。例如，要作如下随机积分：

$$E[f(\mathbf{x},\xi)] = \int_D f(\mathbf{x},\xi)\phi(\xi)\, d\xi$$

对于固定向量 $\mathbf{x}$，令 Y 是随机变量 $|D|\ f(\mathbf{x},\Xi)\phi(\Xi)$，其中 Ξ 是有界域 $D \in R^m$ 上均匀

分布的随机变量，$|D|$是有界域的大小。容易证明 $E[Y]$就是积分的值。用以下简化方法可以计算 $E(Y)$：

$$\overline{Y}(n)=\frac{|D|}{n}\sum_{i=1}^{n}f(x,\Xi_i)\phi(\Xi_i)$$

其中，$\Xi_1,\Xi_2,\cdots,\Xi_n$ 是有界域上均匀分布的独立随机变量；$\overline{Y}(n)$是积分值的无偏估计，且 $\mathrm{Var}[\overline{Y}(n)]=\mathrm{Var}[Y]/n$。对比采用迭代法的传统多元积分，Monte Carlo 仿真与保证精度要求的采样点数和维数无关，这就使得 Monte Carlo 仿真对于高维问题很有吸引力。

Monte Carlo 仿真过程

begin

objective← 0；

for i ← 1 **to** *number_simulation* **do**

  Ξ_i←*random_vector*()；

  objective← *objective*+$f(\mathbf{x},\Xi_i)\phi(\Xi_i)$；

end

objective←*objective*× |D|/*number_simulation*；

end

Monte Carlo 仿真的更详细的讨论可见文献 Hammersley 和 Handscombe[205]，Halton[203]，Rubinstein[361]，以及 Morgan[298]。

2.3.3 随机优化问题的进化程序

Gen，Liu 和 Ida 提出了以下随机优化问题的进化算法[163]：

$$\begin{aligned}&\max\quad E[f(\mathbf{x},\xi)]\\&\text{s.t.}\quad g_i(\mathbf{x})\leqslant 0;\quad i=1,2,\cdots,m\end{aligned}$$

他们称之为进化规划。进化规划的概念最先是由 Michalewicz 提出的[287]。它完全基于遗传算法，其差别只是进化规划主张将适于问题的数据结构和整套有用的遗传算子结合使用。

（1）基因表达与初始化

对于数值优化问题，浮点表达比起二进制表达有许多优点[287]。按浮点表达，每个染色体编码为实数向量，且与决策变量等长。对于有 n 个决策变量的优化问题，可用向量$(x_1,x_2,\cdots,x_n)$作为表达问题解的染色体。

初始化过程应首先确定约束集合中的一个内点 V_0 和一个大的正数 M_0，然后按以下步骤产生 *pop_size* 个染色体：

步骤 1：在 R^n 中随机选择一个方向 $\mathbf{d}$。

步骤 2：令 $M=M_0$. 若 $\mathbf{v}_0+M\cdot\mathbf{d}$ 可行，令其为新的染色体；否则令 M 为$(0, M)$间的随机数，直到 $\mathbf{v}_0+M\cdot\mathbf{d}$ 可行。

步骤 3：重复上述步骤 *pop_size* 次，即产生 *pop_size* 个初始可行解。

由于 V_0 是内点，随机数 M 在迭代中是递减的，所以通过步骤 2 的有限次迭代就可以找到一个可行解。

（2）评估函数

Goldberg 指出[171]：当采用适值正比繁殖时，在算法初始阶段，少数超常个体在一

代中会占用有限种群的过大比例，造成过早收敛，这是不希望发生的。而在后期又会造成种群的分散，并且种群的平均适值可能接近最好适值，这就使得最好染色体的繁衍成了无效的随机游走。为克服这个问题，一些标定方法，比如线性适值标定，幂律标定等方法已经提出[287]。Gen，Liu 和 Ida 提出一种指数-适值标定方法，它基本上是一种排序与标定结合的方法。

首先，用 Monte Carlo 仿真估算每个染色体的目标函数值，然后将所有染色体按目标函数值以从坏到好（即最坏的在位置 1，最好的在位置 *pop_size*）的顺序排序。对于极大问题，即按目标值的升序排序，对极小问题，则按降序排序。

第二，定义三个偏好参数 p_1，p_0 和 p_2（$0<p_1<p_0<p_2<1$），分别用来确定三个标准染色体 u_1，u_0 和 u_2，使得 $u_1=\llcorner p_1\cdot pop_size\lrcorner$，$u_0=\llcorner p_0\cdot pop_size\lrcorner$ 和 $u_2=\llcorner p_2\cdot pop_size\lrcorner$。符号 $\llcorner x\lrcorner$ 表示大于 x 的最小整数。

最后，令第 u_1 位的染色体的适值为 $e^{-1}\approx 0.37$，第 u_0 位的染色体的适值为 1，而第 u_2 位的染色体的适值为 $2-e^{-1}\approx 1.63$。对于第 u 位的染色体 $\mathbf{v}$，它的指数适值 $eval(\mathbf{v})$ 和位数 u 的关系如下：

$$eval(\mathbf{v})=\begin{cases}\exp\left[-\dfrac{u-u_0}{u_1-u_0}\right], & u<u_0\\ 2-\exp\left[-\dfrac{u-u_0}{u_2-u_0}\right], & u\geqslant u_0\end{cases}\tag{2.31}$$

该关系见图 2.4。

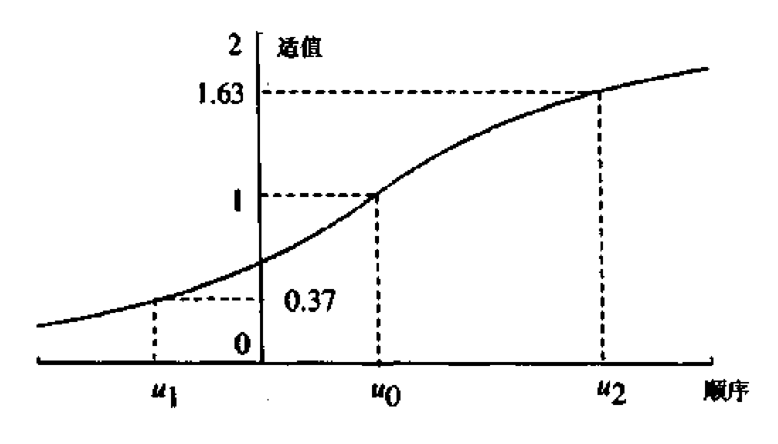


图 2.4 指数适值

然后，按适值 $eval(\mathbf{v}_k)$，$k=1,2,\cdots,pop_size$ 构造转轮，并用来产生新的种群。

(3) 交叉

这里采用算术交叉，即取两个向量的凸组合[287]。如果约束集是凸的，当双亲可行时，这种交叉方法可保证两个后代都是可行的。对于双亲 $\mathbf{v}_1$ 和 $\mathbf{v}_2$，交叉产生的两个后代 $\mathbf{v}'$ 和 $\mathbf{v}''$ 如下：

$$\mathbf{v}'=c_1\cdot\mathbf{v}_1+c_2\cdot\mathbf{v}_2$$
$$\mathbf{v}''=c_2\cdot\mathbf{v}_1+c_1\cdot\mathbf{v}_2$$

其中，$c_1,c_2\geqslant 0$ 且 $c_1+c_2=1$。

(4) 变异

和初始化步骤一样，变异运算按自由方向变异染色体。令父亲为$\mathbf{v}=[x_1,x_2,\cdots,x_n]$，随机产生的变异方向为$\mathbf{d}$，后代即为

$$\mathbf{v}' = \mathbf{v} + M \cdot \mathbf{d}$$

若后代不可行，则在$(0, M)$间再产生随机数M，直到$\mathbf{v}+M\cdot\mathbf{d}$可行为止。

(5) 算法

随机优化问题的进化程序总结如下：

步骤 1：设定参数

最大代数：*max_gen*；

种群大小：*pop_size*；

交叉率：p_c；

变异率：p_m；

偏好参数：p_1, p_0, p_2；

一个大正数：M_0；

当前代数：$gen \leftarrow 0$；

步骤 2：初始化

产生一个内点 $\mathbf{v}_0$；

for $k \leftarrow 1$ **to** *pop_size* **do**

  $M \leftarrow M_0$；

  产生一个随机方向 $\mathbf{d}$；

  $\mathbf{v}_k \leftarrow \mathbf{v}_0 + M \cdot \mathbf{d}$；

  while $\mathbf{v}_k$ 不可行 **do**

    $M \leftarrow random(M)$；

    $\mathbf{v}_k \leftarrow \mathbf{v}_0 + M \cdot \mathbf{d}$；

  end

end

步骤 3：评估

for $k \leftarrow 1$ **to** *pop_size* **do**

  对 $\mathbf{v}_k$ 用 Monte Carlo 仿真计算目标值 f_k；

end

按目标值对染色体排序；

for $k \leftarrow 1$ **to** *pop_size* **do**

  基于位置顺序计算指数适值 $eval(\mathbf{v}_k)$；

end

步骤 4：选择运算

for $k \leftarrow 1$ **to** *pop_size* **do**

  计算选择概率 $p_k = eval(\mathbf{v}_k) \Big/ \sum_{j=1}^{pop_size} eval(\mathbf{v}_j)$；

end

for $k \leftarrow 1$ **to** *pop_size* **do**

```
    计算累计概率 q_k = Σ_{j=1}^{k} p_j ;
  end
  for k←1 to pop_size do
    if q_{k-1}<random()≤q_k then
      选择 v_k;
    end
  end
  步骤 5：交叉运算
  for k←1 to pop_size/2 do
    if random()≤ p_c then
      j ← random(pop_size);
      l ← random(pop_size);
      α ← random();
      v' ← αv_j+ (1−α)v_l;
      v'' ← αv_l+ (1−α) v_j;
    end
  end
  步骤 6：变异运算
  for k ← 1 to pop_size do
    if random()≤ p_m then
      M ← M_0;
      产生一个随机方向 d;
      v'_k←v_k+ M · d;
      whilev_k 不可行 do
        M ← random(M);
        v'_k←v_k+M · d;
      end
    end
  end
  步骤 7：终止条件检查
  gen ← gen +1;
  if gen < max_gen then
    转步骤 3;
  else
    停止;
  end
```

其中 *random*()表示产生一个(0,1)间的随机实数，而 *random*(*num*)表示产生一个(0,*num*)间的随机实数。

例 2.6 考虑如下三个决策变量和三个随机变量的随机优化问题：

$$\max \quad E[f(\mathbf{x},\xi)] = \int_{R^3}[(x_1-\xi_1)\cdot\sin(\pi x_1)+(x_2-\xi_2)\cdot\sin(4\pi x_2)$$
$$+(x_3-\xi_3)\cdot\sin(10\pi x_3)]\phi(\xi_1,\xi_2,\xi_3)d\xi_1 d\xi_2 d\xi_3$$
$$\text{s.t.} \quad 0 \leqslant x_i \leqslant 5;\ i=1,2,3$$

式中，$\phi(\xi_1,\xi_2,\xi_3)$是正态分布的密度函数，其形式如下：

$$\phi(\xi_1,\xi_2,\xi_3)=(2\pi)^{-\frac{3}{2}}(\det C)^{\frac{1}{2}}\exp\left\{-\frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}c_{ij}(\xi_i-u_i)(\xi_j-u_j)\right\} \tag{2.32}$$

其中，$u_1=1,u_2=2,u_3=3$；正定矩阵 $C=(c_{ij})$为

$$c_{ij}=\begin{cases}2, & i=j\\ 0, & i\neq j\end{cases} \tag{2.33}$$

且 $\det C$ 是 C 的行列式的值。遗传算法算得的最好解为$[x_1,x_2,x_3]=[4.5065,4.6279,0.1459]$，目标值是 8.9906。

2.4 非线性目标规划

近年来，遗传算法以其解多目标优化问题的潜力而受到极大的重视。Schaffer 做了该领域先驱性的实验，提出了所谓向量评估遗传算法(VEGA)[369]。Tamaki，Mori 和 Araki 提出了 VEGA 的 Pareto 解方法[396]。Horn，Hafpliotis 和 Goldberg 提出了嵌入 *Pareto* 遗传算法[223]。Fonseca 和 Fleming 描述了多目标遗传算法的基于顺序的适值指定方法[135]。Osyczka 和 Kundu 提出了多目标遗传算法的距离法[324]。Tanino，Tanaka 和 Hojo 提出了一种多目标遗传算法的交互方法[459]。

Sakawa，Kato 和 Shibano 将遗传算法和模糊规划技术结合起来解多目标 0-1 规划[364,455,456]。Gen 和 Liu 则研究了遗传算法在解非线性目标规划中的应用[162]。

目标规划是一种解多目标优化问题的有用技术。目标规划(GP)的概念是 Charnes 和 Cooper 提出的[55]。以后许多研究者都对这个领域作了一些工作，包括 Ijiri[230]，Lee[268]，Ignizio[329] 以及 Gen 和 Ida[153,154]。目标规划的基本思想是对每个目标指定一个希望的目标值，建立各个目标的目标函数，然后找出一个解，使得这些目标函数与目标值的偏差的(加权)和达到极小。目标规划在实际生活中获得了广泛的应用。

有两类目标规划问题。一类是无优先目标规划，它的目标大致同等重要。另一类是优先目标规划，它的目标有不同的优先级。这样，最重要的目标受到第一优先的重视，第二等重要的目标受到第二优先的重视，以此类推。今天目标规划通常指这类优先型的目标规划，它已是多目标决策问题的广为人知的基本方法。

优先非线性目标规划，或简称非线性目标规划是一类包括非线性目标和非线性约束的非常重要的工程设计问题。Ignizio [228]，Hwang 和 Masud[224] 以及 Weistroffer[412] 都讨论过非线性目标规划。Saber 和 Ravindran 还综述和讨论过非线性目标规划的四种主要方法[362]：

(1) 基于单纯形的方法；

(2) 直接搜索方法；

(3) 梯度搜索方法；

(4) 交互方法。

由于非线性目标规划具有不同的结构和非线性度，非线性目标规划技术的性能随要解问题的可靠性、精度、收敛性以及计算和准备的努力程度而异。采用何种非线性目标规划技术取决于要解的问题。

2.4.1 非线性目标规划的公式化

非线性目标规划是一种为了求解具有矛盾的非线性目标和非线性约束的问题，而发展起来的数学规划技术，其中含有用户提供的各目标希望达到的值和目标的优先级和顺序。它要找一个最优解使之按指定的顺序尽可能多地满足这些目标。非线性目标规划的一般公式如下：

$$\min \quad z_0 = \sum_{k=1}^{q} \sum_{i=1}^{m_0} P_k (w_{ki}^{+} d_i^{+} + w_{ki}^{-} d_i^{-}) \tag{2.34}$$

$$\text{s.t.} \quad f_i(\mathbf{x}) + d_i^{-} - d_i^{+} = b_i; \qquad i = 1,2,\cdots,m_0 \tag{2.35}$$

$$g_i(\mathbf{x}) \leqslant 0; \qquad i = m_0 + 1,\cdots,\overline{m}_1 (= m_0 + m_1) \tag{2.36}$$

$$h_i(\mathbf{x}) = 0; \qquad i = \overline{m}_1 + 1,\cdots,m (= \overline{m}_1 + m_2) \tag{2.37}$$

$$d_i^{-},\ d_i^{+} \geqslant 0; \qquad i = 1,2,\cdots,m_0 \tag{2.38}$$

其中：

P_k——第 k 个优先级 ($P_k \gg P_{k+1}$，所有 k)；

d_i^{+}——代表目标 i 过剩量的正偏差变量；

d_i^{-}——代表目标 i 不足量的负偏差变量；

w_{ki}^{+}——指定给 d_i^{+} 的优先级为 P_k 的正权重；

w_{ki}^{-}——指定给 d_i^{-} 的优先级为 P_k 的负权重；

$\mathbf{x}$——n 维决策向量；

f_i——$R^n \to R^1$ 的目标约束函数；

g_i——$R^n \to R^1$ 的实不等式约束函数；

h_i——$R^n \to R^1$ 的实等式约束函数；

b_i——目标 i 的渴望水平和目标值；

q——优先级数；

m_0——目标约束数；

m_1——实不等式约束数；

m_2——实等式约束数。

有时我们将目标函数(2.38)写为

$$\text{lexmin} \quad \left\{ z_1 = \sum_{i=1}^{m} (w_{1i}^{+} d_i^{+} + w_{1i}^{-} d_i^{-}),\cdots,z_q = \sum_{i=1}^{m} (w_{qi}^{+} d_i^{+} + w_{qi}^{-} d_i^{-}) \right\} \tag{2.39}$$

其中，lexmin 表示按字典序极小化目标。

2.4.2 非线性目标规划的遗传算法

Gen 和 Liu 用遗传算法来解非线性目标规划问题[162]，因为遗传算法属于问题独立

的概率算法，且能把握任何种类的目标和约束。由于它的进化性质，遗传算法可以通过维持一个潜在解的种群进行复杂空间中的多方向的鲁棒搜索。所以我们相信它有能力处理实际中的非常复杂的非线性目标规划问题。

(1) 基因表达与初始化

对于 n 个决策变量的给定问题，用向量$[x_1,x_2,\cdots,x_n]$作为表达问题解的染色体。当等式约束 $h_i(\mathbf{x})=0$，$i=1,\cdots,r$ 为线性时，可以容易地用余下的变量消去 r 个变量。这样，染色体变为$[x_1,x_2,\cdots,x_{n-r}]$。

初始化步骤从指定约束集内的一个内点 $\mathbf{v}_0$ 和一个大的正数 M_0 开始。并按以下步骤产生初始的 *pop_size* 个染色体：

步骤 1：在 R^n 中随机地选择一个方向 $\mathbf{d}$。

步骤 2：令 $M=M_0$. 若 $\mathbf{v}_0+M\cdot\mathbf{d}$ 可行，取其为新染色体；否则，令 M 为$(0, M)$中的随机数，直到 $\mathbf{v}_0+M\cdot\mathbf{d}$ 可行为止。

步骤 3：重复步骤 2 *pop_size* 次，从而产生 *pop_size* 个初始可行解。

由于 $\mathbf{v}_0$ 是内点，随机数 M 是递减的，不等式约束的可行解一定能通过步骤 2 的有限步迭代获得。

(2) 评估函数

一类基于排序的评估函数可用来表述每个染色体的优点。评估过程由以下三步构成：

步骤 1：按目标(2.39)计算目标值. 每个染色体含有 q 个目标值，即

$$\left\{\sum_{i=1}^{m_0}(w_{1i}^+d_i^+ + w_{1i}^-d_i^-),\sum_{i=1}^{m_0}(w_{2i}^+d_i^+ + w_{2i}^-d_i^-),\cdots,\sum_{i=1}^{m_0}(w_{qi}^+d_i^+ + w_{qi}^-d_i^-)\right\}$$

步骤 2：按第一优先的目标值 $\sum_{i=1}^{m_0}(w_{1i}^+d_i^+ + w_{1i}^-d_i^-)$ 对染色体排序。若某些染色体有相同目标值，则再按第二目标值 $\sum_{i=1}^{m_0}(w_{2i}^+d_i^+ + w_{2i}^-d_i^-)$ 对它们排序，以此类推。这里，是按目标值的升序排序的。

步骤 3：给每个染色体指定一个基于位置的适值。令 r_k 是染色体 $\mathbf{v}_k$ 在顺序中的位置。对于用户给定的参数 $a\in(0,1)$，基于位置的适值函数定义为

$$eval(\mathbf{v}_k)=a(1-a)^{r_k-1}$$

其中，$r_k=1$ 表示最好的染色体，$r_k=pop_size$ 表示最坏的染色体。因此我们有

$$\sum_{k=1}^{pop_size} eval(\mathbf{v}_k)\approx 1$$

选择过程是转动转轮 *pop_size* 次，每次选出一个染色体进入新种群。

(3) 交叉

这里采用定义为两个向量凸组合的算术交叉[287]。如果约束集凸，可行的双亲将产生可行的后代。对于双亲 $\mathbf{v}_1$ 和 $\mathbf{v}_2$，交叉运算产生的后代 $\mathbf{v}'$ 和 $\mathbf{v}''$如下：

$$\mathbf{v}'=c_1\cdot\mathbf{v}_1+c_2\cdot\mathbf{v}_2$$
$$\mathbf{v}''=c_2\cdot\mathbf{v}_1+c_1\cdot\mathbf{v}_2$$

这里，$c_1,c_2\geqslant 0$ 且 $c_1+c_2=1$。

(4) 变异

变异运算采用和初始化步骤同样的沿自由方向变异染色体的方法。令父亲为 $\mathbf{v}=[x_1,x_2,\cdots,x_n]$,随机产生的方向为 $\mathbf{d}$。变异产生的后代如下：

$$\mathbf{v}' = \mathbf{v} + M \cdot \mathbf{d}$$

若后代不可行,则令 M 为$(0, M)$间的随机数,直到 $\mathbf{v}+ M \cdot \mathbf{d}$ 可行为止。

2.4.3 数值例子

本节的例子都采用以下参数设定:种群大小 $pop_size=30$,交叉概率 $p_c=0.2$,变异概率 $p_m=0.6$,位置评估函数中的参数 $a=0.1$。

例 2.7 第一个例子源于 Ignizio[228],其问题为

$$\begin{aligned}
\text{lexmin} \quad & \{d_3^+, 2d_1^- + d_2^+\} \\
\text{s.t.} \quad & x_1 x_2 + d_1^- - d_1^+ = 16 \\
& (x_1 - 3)^2 + x_2^2 + d_2^- - d_2^+ = 9 \\
& x_1 + x_2 + d_3^- - d_3^+ = 6
\end{aligned}$$

用遗传算法运行 300 代获得的最好解为

$$\mathbf{x} = [3.01858, 2.98136]$$

它很接近于已知解 (3,3),前者的目标值是 $d_3^+=0$ 和 $2d_1^-+d_2^+=14.0010$,而后者的目标值为 $d_3^+=0$ 和 $2d_1^-+d_2^+=14.0000$。

例 2.8 第二个例子是 Van de Panne 和 Popp 提出的[408],Lee 和 Olson 也解过这个问题[269]。这是一个牛饲料混合问题,要混合四种饲料以满足蛋白质和脂肪的要求。其问题为

$$\begin{aligned}
\text{lexmin} \quad & \{d_1^- + d_2^- + d_3^- + d_4^-, d_5^-, d_6^+, d_7^-, d_8^+, d_9^-, d_{10}^+, d_{11}^-, d_{12}^+\} \\
\text{s.t.} \quad & x_1 + d_1^- - d_1^+ = 0.01 \\
& x_2 + d_2^- - d_2^+ = 0.01 \\
& x_3 + d_3^- - d_3^+ = 0.01 \\
& x_4 + d_4^- - d_4^+ = 0.01 \\
& 12.0x_1 + 11.9x_2 + 41.8x_3 + 52.1x_4 \\
& \quad - 0.841(0.2809x_1^2 + 0.1936x_2^2 + 20.25x_3^2 + 0.6241x_4^2)^{\frac{1}{2}} + d_5^- - d_5^+ = 21 \\
& 24.55x_1 + 26.75x_2 + 39.00x_3 + 40.50x_4 + d_6^- - d_6^+ = 30 \\
& 12.0x_1 + 11.9x_2 + 41.8x_3 + 52.1x_4 \\
& \quad - 1.282(0.2809x_1^2 + 0.1936x_2^2 + 20.25x_3^2 + 0.6241x_4^2)^{\frac{1}{2}} + d_7^- - d_7^+ = 21 \\
& 24.55x_1 + 26.75x_2 + 39.00x_3 + 40.50x_4 + d_8^- - d_8^+ = 29.90 \\
& 12.0x_1 + 11.9x_2 + 41.8x_3 + 52.1x_4 \\
& \quad - 1.645(0.2809x_1^2 + 0.1936x_2^2 + 20.25x_3^2 + 0.6241x_4^2)^{\frac{1}{2}} + d_9^- - d_9^+ = 21 \\
& 24.55x_1 + 26.75x_2 + 39.00x_3 + 40.50x_4 + d_{10}^- - d_{10}^+ = 29.80 \\
& 12.0x_1 + 11.9x_2 + 41.8x_3 + 52.1x_4 \\
& \quad - 2.323(0.2809x_1^2 + 0.1936x_2^2 + 20.25x_3^2 + 0.6241x_4^2)^{\frac{1}{2}} + d_{11}^- - d_{11}^+ = 21 \\
& 24.55x_1 + 26.75x_2 + 39.00x_3 + 40.50x_4 + d_{12}^- - d_{12}^+ = 0
\end{aligned}$$

$$x_1 + x_2 + x_3 + x_4 = 1$$
$$2.3x_1 + 5.6x_2 + 11.1x_3 + 1.3x_4 \geqslant 5$$
$$x_1, x_2, x_3, x_4 \geqslant 0$$

等式约束 $x_1+x_2+x_3+x_4=1$ 可用 $1-(x_1+x_2+x_3)$ 替代 x_4。这样，染色体可表达为 $[x_1, x_2, x_3]$。初始化需要的内点给定为[0.2,0.3,0.4]。用遗传算法运行 600 代获得的最好解为

$$\mathbf{x} = [0.62742, 0.01001, 0.30914, 0.05343]$$

它非常接近于 Lee 和 Olson 报告的解 $\mathbf{x}=[0.62743, 0.01000, 0.30914, 0.05343)]$[269]。

例 2.9 本例来源于 El-Sayed，Ridgely 和 Sandgren 的工作[118]，其问题为

$$\begin{aligned}
\min \quad & z = d_1^- + d_2^- + d_2^+ + d_3^- + d_3^+ + d_4^- + d_4^+ \\
\text{s.t.} \quad & 5.28x_1^2 + 3.74x_2^2 + 5.28x_3^2 + d_1^- - d_1^+ = 3.575 \\
& 0.178/x_1^2 + d_2^- - d_2^+ = 1.0 \\
& 0.255/x_2^2 + d_3^- - d_3^+ = 1.0 \\
& 0.178/x_3^2 + d_4^- - d_4^+ = 1.0 \\
& x_1, x_2, x_3 \geqslant 0
\end{aligned}$$

这是一个三棒结构的优化问题。El-Sayed 等的算法采用了将非线性方程分段线性化，再用线性目标规划求解的方法，获得的解为 $\mathbf{x}=[0.4239, 0.5047, 0.4239]$，最小目标值为 $z=0.725$。

用遗传算法运行 300 代得到的解为

$$\mathbf{x} = [0.4219, 0.6733, 0.4219], \quad z_0 = 0.4376$$

它比 Ei-Sayed 报告的结果[118]好得多。

例 2.10 下面考虑一个复杂的非线性目标规划的例子：

$$\begin{aligned}
\text{lexmin} \quad & \{z_1 = d_1^-, z_2 = d_2^-, z_3 = d_3^+, z_4 = d_4^- + d_4^+\} \\
\text{s.t.} \quad & x_1\sin(\pi x_1) + x_2\sin(2\pi x_2) + x_3\sin(3\pi x_3) + x_4\sin(4\pi x_4) \\
& \quad + x_5\sin(5\pi x_5) + d_1^- - d_1^+ = 18 \\
& x_1\sin(\pi x_1) + x_2\sin(2\pi x_2) + x_3\sin(3\pi x_3) + x_4\sin(4\pi x_4) \\
& \quad + d_2^- - d_2^+ = 15 \\
& x_1\sin(\pi x_1) + x_2\sin(2\pi x_2) + x_3\sin(3\pi x_3) + d_3^- - d_3^+ = 10 \\
& x_1\sin(\pi x_1) + x_2\sin(2\pi x_2) + d_4^- - d_4^+ = 0 \\
& x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \leqslant 100
\end{aligned}$$

该例子比前面的目标规划例子复杂，因为它的约束函数是多模态的，其中一项 $x\sin(5\pi x)$ 如图 2.5 所示。带有 $x\sin(i\pi x)$ 和的函数非常复杂，常规方法无法求解。然而遗传算法对这类目标规划问题却是有效的。用遗传算法运行 6000 代获得的最好解为

$$\mathbf{x} = [-1.860, 0.745, 6.823, 6.685, 1.995]$$

四个目标值随代数变化的图形见图 2.6。

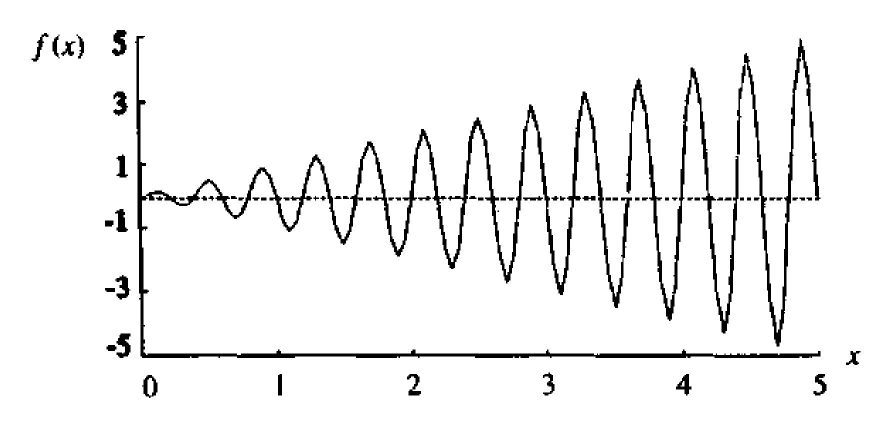


图 2.5 函数 $f(x)=x\sin(5\pi x)$的曲线

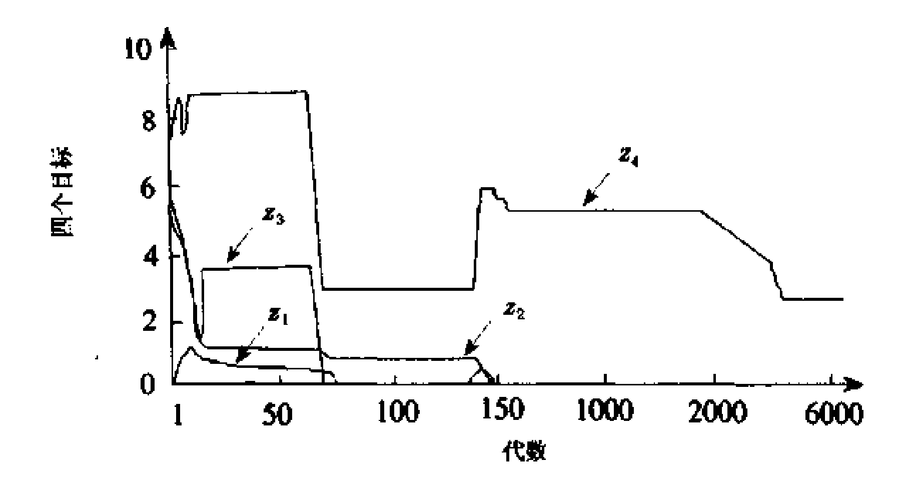


图 2.6 问题评估中的四个目标

2.5 区间规划

过去 10 年，提出了两种不同的区间优化的方法：

(1) 区间变量和正常系数的问题；

(2) 区间系数和正常变量的问题。

区间分析是为第一类问题提出的[207]，采用区间分析和遗传算法结合的混合方法来求全局最优解的工作已见诸报道[4,306]。区间规划技术是为解决第二类问题发展起来的[232,311]。当使用数学规划方法来解实际问题时，对于决策者来说往往很难确定模型的参数值。但是，这种不确定性却可以粗略地用置信区间来描述，这便促进了区间规划的发展。

本节将说明如何用遗传算法解区间规划问题。基本思想是首先将区间规划转化为双目标规划模型，再用遗传算法找该模型的 Pareto 解。

2.5.1 引言

本小节介绍区间算术的基本定义、不等式成立的度、区间之间的顺序关系以及一些相关的定理。

1. 区间算术

区间被定义为一对有序的实数，例如[6]：

$$A=[a^L,a^R] \\ =\{x|a^L\leqslant x\leqslant a^R;\ x\in R^1\} \tag{2.40}$$

式中，a^L 和 a^R 分别是区间 A 左界和右界。区间还可按如下方式定义：

$$A=<a^C,a^W> \\ =\{x|a^C-a^W\leqslant x\leqslant a^C+a^W;\ x\in R^1\} \tag{2.41}$$

式中，a^C 和 a^W 分别是区间 A 的中心和宽度。它们可按下式计算：

$$a^C=\frac{1}{2}(a^R+a^L) \tag{2.42}$$

$$a^W=\frac{1}{2}(a^R-a^L) \tag{2.43}$$

图 2.7 给出了区间 A 的说明。

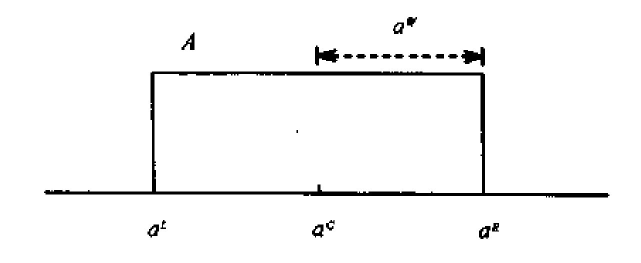


图 2.7　区间的说明

将普通算术扩展到封闭区间就得到区间算术。对于两个区间 $A=[a^L,a^R]$ 和 $B=[b^L,b^R]$，区间算术的基本定义如下[6,207]：

$$A+B=[a^L+b^L,a^R+b^R] \tag{2.44}$$

$$A-B=[a^L-b^R,a^R-b^L] \tag{2.45}$$

$$kA=\begin{cases}[ka^L,ka^R], & k\geqslant 0\\ [ka^R,ka^L], & k<0\end{cases} \tag{2.46}$$

$$A\times B=[a^L\times b^L,a^R\times b^R],\quad a^L\geqslant 0,b^L\geqslant 0 \tag{2.47}$$

$$\frac{A}{B}=\left[\frac{a^L}{b^R},\frac{a^R}{b^L}\right],\quad a^L\geqslant 0,b^L\geqslant 0 \tag{2.48}$$

$$\log(AB)=\log(A)+\log(B),\quad a^L\geqslant 0,b^L\geqslant 0 \tag{2.49}$$

运算(2.49)的证明可在文献[312]中找到。

2. 区间不等式

考虑带有区间系数的以下约束：

$$\sum_{j=1}^{n}A_jx_j\leqslant B \tag{2.50}$$

式中

$$B=[b^L,b^R],A_j=[a_j^L,a_j^R],x_j\geqslant 0;\quad j=1,\cdots,n \tag{2.51}$$

如何确定约束的可行域是区间规划的基本问题。过去几年中已提出了几种可行域的定义。Ishibuchi 和 Tanaka 的定义是根据两个区间不等式成立的度的概念提出的[231]。

定义 2.1 对于区间 A 和实数 x，不等式 $A \leqslant x$ 成立的度定义如下：

$$g(A \leqslant x) = \max\left\{0, \min\left\{1, \frac{x - a^L}{a^R - a^L}\right\}\right\} \tag{2.52}$$

该定义可用图 2.8 来说明。

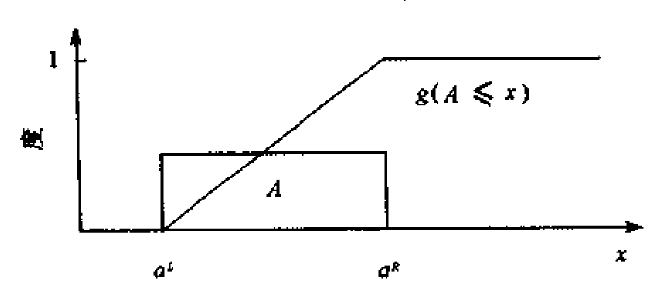


图 2.8 不等式成立的度的说明

定义 2.2 对于两个区间 A 和 B，不等式 $A \leqslant B$ 成立的度定义如下：

$$g(A \leqslant B) = g(A - B \leqslant 0) = \max\left\{0, \min\left\{1, \frac{b^R - a^L}{a^R - b^L + b^R - a^L}\right\}\right\} \tag{2.53}$$

按照定义 2.2，区间约束(2.49)的可行域可用如下定理来确定：

定理 2.1 (Ishibuchi 和 Tanaka[231]) 对于一个给定的不等式成立的度 q，区间不等式约束(2.50)可以转换为如下清晰的不等式约束：

$$\sum_{j=1}^{n} (q\, a_j^R x_j + (1 - q)\, a_j^L x_j) \leqslant (1 - q)\, b^R + q\, b^L \tag{2.54}$$

区间约束的可行域的更详细的讨论见文献[311]。

3. 区间之间的顺序关系

考虑如下区间规划问题：

$$\max\left\{Z(\mathbf{x}) = \sum_{j=1}^{t} C_j x_j \mid \mathbf{x} \in S \subset R_+^n\right\} \tag{2.55}$$

式中，S 是 $\mathbf{x}$ 的可行域；C_j 代表 x_j 的不确定的单位利益的区间系数。对于给定的 $\mathbf{x}$，总的利益 $Z(\mathbf{x})$ 是一个区间数。我们需要根据这个区间利益来作决策。区间之间的顺序就是为此提出的，它表达了决策者对区间利益的偏好。以下区间顺序关系是为极大化问题定义的：

定义 2.3 对于两个区间 A 和 B，顺序关系 $\leqslant_{LR}$ 定义如下：

$$A \leqslant_{LR} B, \quad 当且仅当\ a^L \leqslant b^L 和\ a^R \leqslant b^R \tag{2.56}$$

该顺序关系表达了决策者对较高的最小利益和较高的最大利益二者的偏好。

定义 2.4 对于区间 A 和 B，顺序关系 $\leqslant_{CW}$ 定义如下：

$$A \leqslant_{CW} B, \quad 当且仅当\ a^C \leqslant b^C 和\ a^W \geqslant b^W \tag{2.57}$$

该顺序关系表达了决策者对较高期望值和较小的不确定性二者的偏好。

定义 2.5 对于区间 A 和 B，顺序关系 $\leqslant_{LC}$ 定义如下：

$$A \leqslant_{LC} B, \quad 当且仅当\ a^L \leqslant b^L 和\ a^C \leqslant b^C \tag{2.58}$$

注意上述顺序关系都是偏序关系，它们是传递的、反射的和反对称的。

根据定义 2.5，问题(2.55)的解集可定义为以下非劣解：

定义 2.6 向量 $\mathbf{x}\in S$ 是问题 (2.55) 的解，当且仅当不存在 $\mathbf{x}'\in S$ 使得

$$Z(\mathbf{x}) \leqslant_{LC} Z(\mathbf{x}') \tag{2.59}$$

按照定义 2.6，区间规划问题 (2.55) 可转换为如下等价的清晰双目标规划问题。

定理 2.2 (Ishibuchi 和 Tanaka[231]) 问题 (2.55) 由定义 2.5 定义的解集可以由如下双目标规划问题的 Pareto 解获得：

$$\max \{z^L(\mathbf{x}),\ z^C(\mathbf{x}) \mid \mathbf{x}\in S\subset R_+^n\} \tag{2.60}$$

下面考虑以下非线性区间规划问题：

$$\max \left\{Z(\mathbf{x}) = \prod_{j=1}^{n} C_j x_j \mid \mathbf{x}\in S\subset R_+^n\right\} \tag{2.61}$$

其中目标为积的形式。由下面的定理这类非线性区间规划问题可以转换为一个等价的线性区间规划问题。

定理 2.3 (Nakahara 和 Gen[312]) 对于给定的顺序关系$\leqslant_{LC}$和两个正的区间 A 和 B，下述关系成立：

$$A \leqslant_{LC} B \Leftrightarrow \log(A) \leqslant_{LC} \log(B) \tag{2.62}$$

推论 1 对于非线性区间规划问题 (2.61)，若 $c_j^L\geqslant 0$，$\forall\ j$，则等价于如下线性区间规划问题：

$$\max \left\{Z(\mathbf{x}) = \sum_{j=1}^{n} \log(C_j x_j) \mid \mathbf{x}\in S\subset R_+^n\right\} \tag{2.63}$$

4. 极大化问题

将区间规划转换为双目标线性规划包括两个关键步骤：

(1) 用两个区间的不等式成立度的定义将区间约束转换为等价的清晰约束；

(2) 用区间之间的顺序关系的定义将区间目标转换为等价的清晰的双目标。

考虑如下极大化区间规划问题：

$$\max \quad Z(\mathbf{x}) = \sum_{j=1}^{n} C_j x_j \tag{2.64}$$

$$\text{s.t.} \quad G_i(\mathbf{x}) = \sum_{j=1}^{n} A_{ij} x_j \leqslant B_i; \quad i = 1,2,\cdots,m \tag{2.65}$$

$$x_j^L \leqslant x_j \leqslant x_j^U,\ \text{整数}; \quad j = 1,2,\cdots,n \tag{2.66}$$

式中，$C_j=[c_j^L,c_j^R]$；$A_{ij}=[a_{ij}^L,a_{ij}^R]$；$B=[b^L,b^R]$；$x_j^L$和 x_j^U 分别为 x_j 的下、上界。问题(2.64)～(2.66)可转换为如下问题：

$$\max \quad z^L(\mathbf{x}) = \sum_{j=1}^{n} c_j^L x_j \tag{2.67}$$

$$\max \quad z^C(\mathbf{x}) = \sum_{j=1}^{n} \frac{1}{2}(c_j^L + c_j^R)\, x_j \tag{2.68}$$

$$\text{s.t.} \quad g_i(\mathbf{x}) = \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i; \quad i = 1,2,\cdots,m \tag{2.69}$$

$$x_j^L \leqslant x_j \leqslant x_j^U,\ \text{整数}; \quad j = 1,2,\cdots,n \tag{2.70}$$

式中，$a_{ij}= q\, a_{ij}^{R}+ (1-q)\, a_{ij}^{L}$；$b_i= (1-q)\, b_i^R+ q\, b_i^L$。

在双目标情况下，两目标通常本质上是相互矛盾的，最优解需要替代为非全优解（或称有效解、Pareto 优解、非劣解），即对于任何目标函数在不牺牲其他目标的情况下就不能改进的解。令 F 为可行解的集合，非全优解的规范定义如下：

定义 2.7 可行解 $\bar{\mathbf{x}} \in F$ 称为非全优解。当且仅当

$$\forall \mathbf{x} \in F, \quad z(\mathbf{x}) \leqslant z(\bar{\mathbf{x}}) \Rightarrow z(\mathbf{x}) = z(\bar{\mathbf{x}}) \tag{2.71}$$

其中

$$z(\mathbf{x}) = [z_1(\mathbf{x}), z_2(\mathbf{x}), \cdots, z_q(\mathbf{x})]^T$$

目标空间中有两个特殊点：正理想解和负理想解。它们代表目标最好和最坏两个极端情况。虽然它们是不可行的，但却能为决策者提供一些建议信息。

定义 2.8 正理想解（PIS）由所有可达到的最好的目标值构成，记为 $z^+=\{z_1^+, z_2^+, \cdots, z_q^+\}$，其中，$z_q^+$ 为不考虑其他目标时第 q 个目标所取得的最好值。

定义 2.9 负理想解（NIS）由所有可达到的最坏的目标值构成，记为 $z^-=\{z_1^-, z_2^-, \cdots, z_q^-\}$，其中，$z_q^-$ 为不考虑其他目标时第 q 个目标所取得的最坏值。

在已有的多目标技术中基本上可分为如下三大类[229]：

(1) 加权法或效用法；

(2) 排序法或优先法；

(3) 有效解法或生成法。

加权法是试图用单一测量来表达所有目标的方法。从计算的观点看它是十分有吸引力的。然而它的明显缺点是实际上常常难于确定真实可信的权重。

排序法或优先法试图绕过上述困难问题。它根据目标的重要性给每个目标指定优先级。多数决策者能够做到这一点。

最后一类方法避免了找权重和优先级的问题。它产生整个非全优解的集合或近似的集合，然后让决策者自己来选择最好地表达他对各个目标的权衡取舍的非全优解。Cohon 称这个解为最好构成解[80]。

2.5.2 遗传算法

Gen 和 Cheng 研究了如何用遗传算法来解区间规划的问题[151]。所提出方法的基本思想是首先将区间规划模型转化为双目标规划模型，然后用遗传算法找出双目标规划的 Pareto 解。

当用遗传算法解多目标问题时，关键的问题是如何评估染色体的信度，并用适值来表达这个信度。他们提出了强迫遗传搜索 Pareto 解集的加权和方法。

1. 染色体表达与初始种群

染色体定义如下：

$$\mathbf{x}^k = [\, x_1^k\ x_2^k \cdots x_n^k]$$

式中，k 是染色体下标。初始种群在$[x_j^L, x_j^U]$所有 x_j 中随机产生。

2. 交叉和变异

交叉用均匀交叉运算实现[391]。已经证明，对于组合优化问题它好于传统的交叉策略。均匀交叉首先产生一个随机的交叉罩，然后对双亲交换交叉罩中的基因。交叉罩是一个和染色体同样位长的二进制串。交叉罩确定的等值的位将使后代的相应位从同一父亲中接受相应位的基因。以上关系可用图 2.9 来说明。

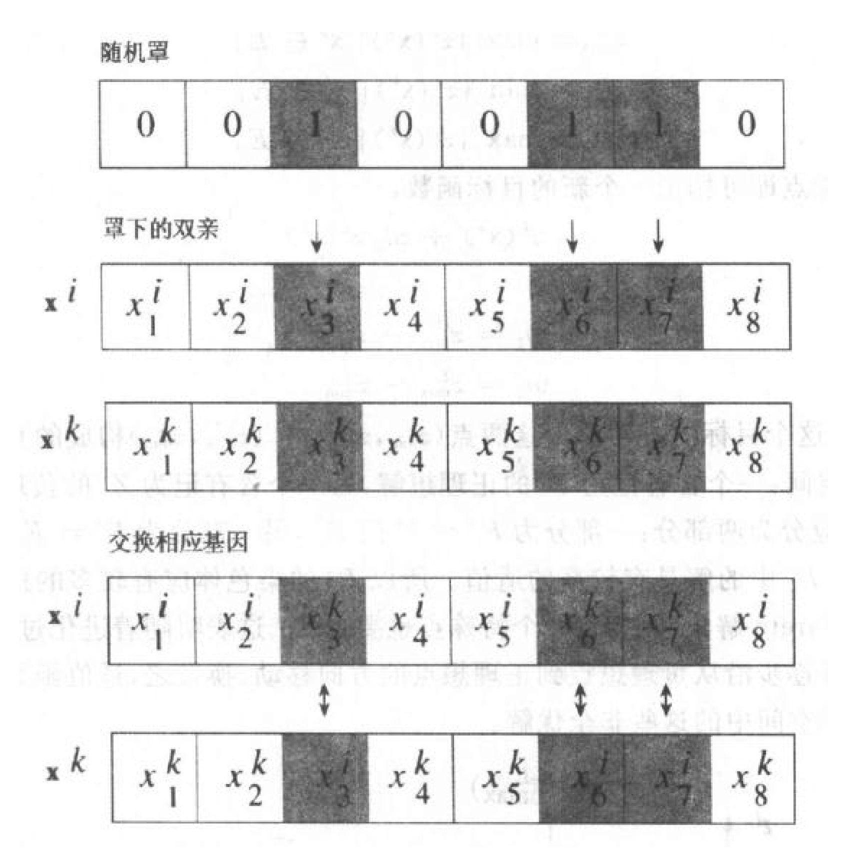


图 2.9 均匀交叉运算的说明

变异采用变量允许范围内的随机摄动。

3. 选择

这里采用确定型选择，即删去双亲和后代中所有重复的染色体，将余下的染色体按降序排列，然后选择前 *pop_size* 染色体作为新的种群。

4. 染色体的评估

这一阶段包括两个主要任务：①如何控制不可行染色体和②如何按双目标确定染色体的适值。令 $\mathbf{x}^k$ 为当前代中的第 k 个染色体，评估函数定义如下：

$$eval(\mathbf{x}^k) = (w_1 z^L(\mathbf{x}^k) + w_2 z^C(\mathbf{x}^k))\, p(\mathbf{x}^k) \tag{2.72}$$

式中，w_1 和 w_2 分别为对应于两个目标重要性的权重。评估函数包含两项：目标的加权和与惩罚。目标的加权和项试图产生选择压力迫使遗传搜索去开发 Pareto 解集，而惩罚项则使遗传搜索从可行域和不可行域两个方面去逼近 Pareto 解。

5. 目标的加权和

与传统遗传算法相比，双目标规划实现的显著特点是它在进化过程中必须保持一个非全优解的集合。E 为已经获得的非全优解集。在 E 中有两个感兴趣的点，一个含有 E 中 $z^L(\mathbf{x}^k)$ 的极大值，另一个含有$z^C(\mathbf{x}^k)$的极大值。记这两点分别为$(z^C_{\min}, z^L_{\max})$和$(z^C_{\max}, z^L_{\min})$，其中

$$z^C_{\min} = \min \{z^C(\mathbf{x}^k) \mid \mathbf{x}^k \in E\}$$
$$z^C_{\max} = \max \{z^C(\mathbf{x}^k) \mid \mathbf{x}^k \in E\}$$
$$z^L_{\min} = \min \{z^L(\mathbf{x}^k) \mid \mathbf{x}^k \in E\}$$
$$z^L_{\max} = \max \{z^L(\mathbf{x}^k) \mid \mathbf{x}^k \in E\}$$

根据这两个特殊点即可构造一个新的目标函数：

$$w_1 z^L(\mathbf{x}^k) + w_2 z^C(\mathbf{x}^k)$$

其中

$$w_1 = z^C_{\max} - z^C_{\min}$$
$$w_2 = z^L_{\max} - z^L_{\min}$$

图 2.10 给出了这个目标的说明。由这两点$(z^C_{\min}, z^L_{\max})$和 $(z^C_{\max}, z^L_{\min})$构成的直线将目标空间分为两个半空间：一个含有记为 Z^+ 的正理想解，另一个含有记为 Z^- 的负理想解。可行解空间 F 也相应分为两部分：一部分为 $F^- = F \cap Z^-$，另一部分为 $F^+ = F \cap Z^+$。容易证明 F^+的解比 F^-中的解具有较高的适值。所以 F^+的染色体应有较多的机会进入下一代。在每代中，Pareto 解集被更新，两个特殊点也要更新。这表明随着进化过程进行，这两点构成的直线将逐步沿从负理想点到正理想点的方向移动。换言之，适值函数将迫使遗传搜索去探索目标空间中的这些非全优解。

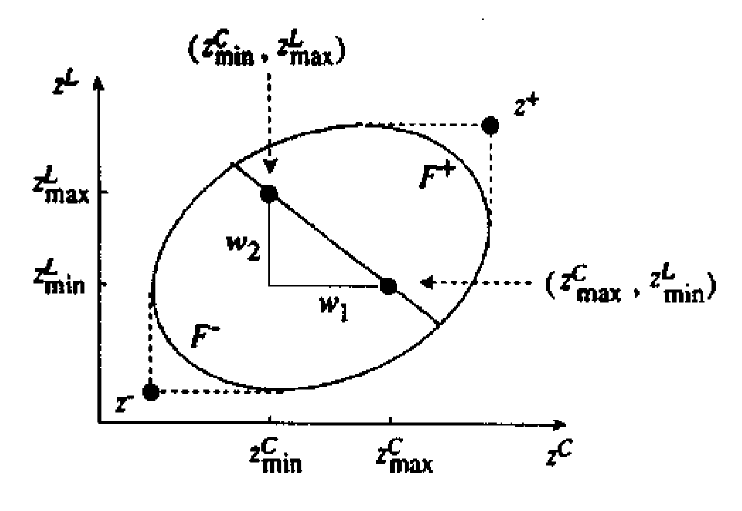


图 2.10 探索新目标的图示

6. 惩罚

第 k 个染色体的惩罚项定义如下：

$$p(\mathbf{x}^k) = 1 - \frac{1}{m}\sum_{i=1}^{m}\frac{\delta(g_{ki})\ (g_{ki} - b_i)}{g_i^{\max} - b_i}$$

其中

$$g_{ki} = \begin{cases} 0, & g_i(\mathbf{x}^k) \leqslant b_i \\ g_i(\mathbf{x}^k), & \text{其他} \end{cases}$$

$$g_i^{\max} = \max\{g_{ki} \mid k = 1,2,\cdots, pop_size\}$$

$$\delta(g_{ki}) = \begin{cases} 0, & g_{ki} = 0 \\ 1, & \text{其他} \end{cases}$$

惩罚项可看成是染色体不可行性的测量。初始化过程或繁殖过程中产生的染色体都可能违反系统约束。这个特殊的测量是用来评估不可行的染色体与可行域的距离的。通常最优点发生在可行域的边界上，当我们给每个不可行的染色体以大的定常的惩罚时，算法在进化中将拒绝所有不可行解，使得遗传搜索只能从可行域一边逼近最优点。这里提出的惩罚方法则可从可行和不可行两边逼近最优解，如图 2.11 所示。

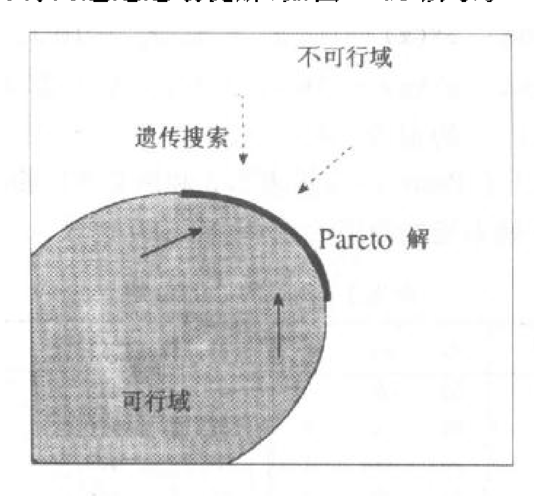


图 2.11　遗传搜索的方向

7. 总的步骤

整个步骤总结如下：

步骤 1：设定参数：设定种群大小 pop_size，变异率 p_m，交叉率 p_c 和最大代数 max_gen。令 $t \leftarrow 0$ 和 $E \leftarrow \varnothing$.

步骤 2：初始化：随机产生初始种群 $\mathbf{x}^k$，$k = 1, 2, \cdots, pop_size$。

步骤 3：交叉：进行均匀交叉。

步骤 4：变异：进行随机摄动变异。

步骤 5：更新集合 E：

1）对每个染色体计算两个目标的值；

2）将新的非全优解加入 E，从而更新 E 并从 E 删去劣点；

3）确定新的特殊点$(z_{\min}^C, z_{\max}^L)$和$(z_{\max}^C, z_{\min}^L)$。

步骤 6：评估：按公式(2.71)计算所有染色体的适值。

步骤 7：选择：

1）删去所有重复的染色体；

2）按降序排列余下的染色体；

3）选择前 pop_size 个染色体组成新的种群。

步骤 8：检查终止条件：若 $t=max_gen$，停止，否则，令 $t \leftarrow t+1$，转步骤 3。

2.5.3 数值例子

考虑如下区间目标函数的例子：

$$\max \quad Z(\mathbf{x}) = [15,17]\,x_1 + [15,20]\,x_2 + [10,30]\,x_3 \tag{2.73}$$

$$\text{s.t.} \quad g_1(\mathbf{x}) = x_1 + x_2 + x_3 \leqslant 30 \tag{2.74}$$

$$g_2(\mathbf{x}) = x_1 + 2\,x_2 + x_3 \leqslant 40 \tag{2.75}$$

$$g_3(\mathbf{x}) = x_2 + 4\,x_3 \leqslant 60 \tag{2.76}$$

$$x_j \geqslant 0,\ \text{整数}; \quad j = 1,2,3 \tag{2.77}$$

该问题可以转换为以下双目标规划问题：

$$\max \quad z^L(\mathbf{x}) = 15\,x_1 + 15\,x_2 + 10\,x_3 \tag{2.78}$$

$$\max \quad z^C(\mathbf{x}) = 16\,x_1 + 17.5\,x_2 + 20\,x_3 \tag{2.79}$$

$$\text{s.t.} \quad \text{约束}(2.73) \sim (2.77)$$

用遗传算法总共找到了 13 个 Pareto 解，见表 2.3 和图 2.12。相应的区间目标见图 2.13。由这幅图我们看到这些区间不能按顺序关系$\leqslant_{LC}$来相互比较。

表 2.3 例子的 Pareto 解

n	$z^L(\mathbf{x})$	$z^C(\mathbf{x})$	x_1	x_2	x_3	n	$z^L(\mathbf{x})$	$z^C(\mathbf{x})$	x_1	x_2	x_3
1	450	492	22	8	0	8	415	520	15	8	7
2	445	496	21	8	1	9	410	524	14	8	8
3	440	500	20	8	2	10	405	531	11	10	9
4	435	505	18	9	3	11	400	533	11	9	10
5	430	508	18	8	4	12	395	534	12	7	11
6	425	510	18	7	5	13	390	543	8	10	12
7	420	519	14	10	6						

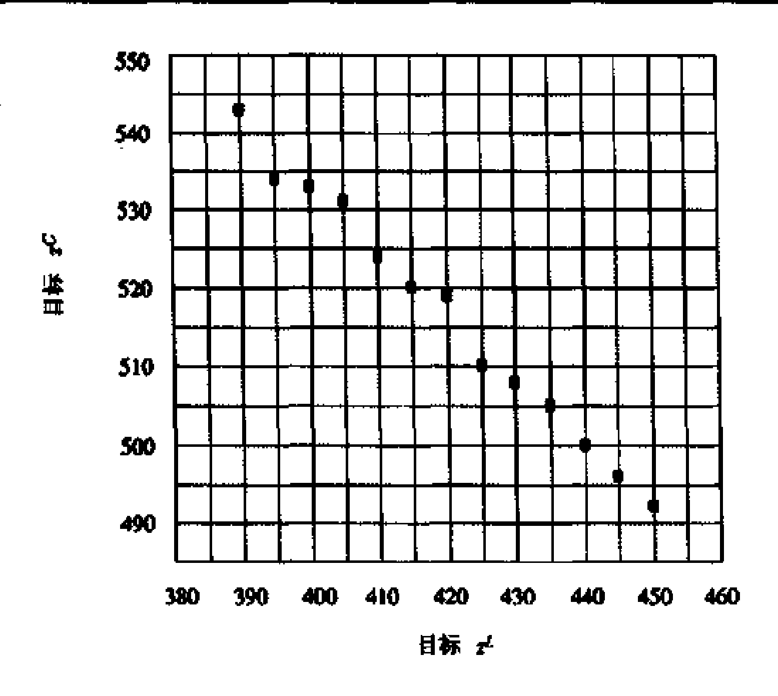


图 2.12 遗传算法获得的 Pareto 解

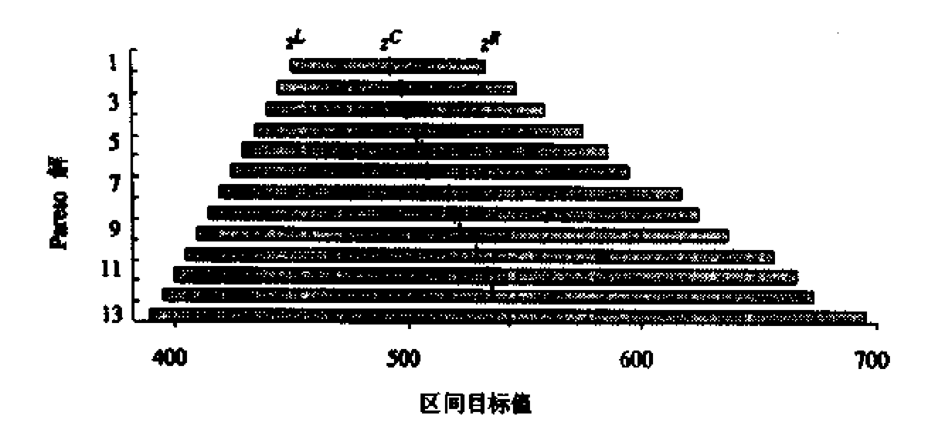


图 2.13　Pareto 解的区间目标

第三章　组合优化问题

3.1 引　言

组合优化研究的是具有有限个可行解的优化问题。在日常生活中，特别是在工程设计中，有许多这样的问题。其中一个重要而广阔的应用领域就是如何有效地利用不充足的资源以提高生产效益。典型的工程设计问题包括背包问题、二次 0-1 整数规划、车辆路径、网络设计、设备分配与布置以及旅行商问题等等。最近，欧洲的一些年轻的研究者对组合优化的研究现状与前景提出了他们的看法[37]。

虽然理论上这类问题的最优解可以用简单的穷举法找到，但实际上这通常是不可行的。特别是实际问题的可行解数量可能非常大。组合优化中最具挑战性的问题就是组合爆炸。解这类难解问题的主要趋势是采用启发式方法，在组合优化中启发式方法明显地起着核心作用。

过去几年里，遗传算法在组合优化学界里日益受到重视，并在许多工业工程的实验与实际应用中取得了良好的成果，显示了它的巨大潜力。由于变型启发式能给用户较大的灵活性，又不要求太多的问题的专门知识就能取得很好的解，因此预计在今后几年里这个领域将会有飞速的发展。

本章将介绍如何用遗传算法解背包问题、二次指派问题、最小生成树问题、旅行商问题以及影片递送问题。所有解这些问题的技术也可以用来解其他组合优化问题。特别是旅行商问题通常作为组合难题的典范，许多新方法都用它来检验。其他问题，比如作业车间调度问题、机器调度问题、车辆路径问题、设备布置问题等，将在以后几章中专门讨论。

3.2 背包问题

前 10 年，背包问题吸引了许多理论和实际工作者，对此问题作了深入的研究。理论研究的兴趣在于尽管问题的结构形式简单，但它却具有组合爆炸的性质，而且许多优化问题都可以通过解一系列背包子问题来解决。从实际的观点看，许多工业问题可以用背包问题来描述，如资金预算，货仓装载，存储分配等，都是典型的应用例子[282,305]。

假设我们要从多种物品(一般称为项目)中选择几件物品，装满背包。若有 n 个不同的项目，对于项目 j，其重量为 w_j，价值为 p_j；W 是背包承受的最大重量。背包问题就是要在不超过背包承重量的前提下，使装入背包的价值最大。这里，价值、重量和承重量都是正整数。

设 x_j 为如下 0-1 变量：

$$x_j = \begin{cases} 1, & \text{项目 } j \text{ 被选入} \\ 0, & \text{其他} \end{cases}$$

背包问题可以用如下数学公式表述：

$$\max \quad \sum_{j=1}^{n} p_j x_j \tag{3.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_j x_j \leq W \tag{3.2}$$

$$x_j = 0 \text{ 或 } 1; \quad j = 1,2,\cdots,n \tag{3.3}$$

这就是 0-1 背包问题，是单约束的纯整数规划，它形成了整数规划的一个重要类别。背包问题有多种变型，比如多选择背包问题、有界背包问题、无界背包问题等。已经证明背包问题是 NP 难问题。已有一些研究者用遗传算法来解背包问题[179,216,321]，提出的遗传算法大略可分为三类：

(1) 二进制表达法；

(2) 顺序表达法；

(3) 变长表达法。

下面我们将分别说明。

3.2.1 二进制表达法

二进制串是 0-1 背包问题的自然表达，其中 1 表示选入，0 表示未选入。例如，一个 10 个项目的背包问题的解可以表示为如下二进制串：

$$\mathbf{x} = [x_1\, x_2 \cdots x_{10}]$$
$$[\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0\,]$$

这表示项目 2，4 和 9 被选入背包。然而这种表达会产生不可行解。换言之，当字串中的 1 太多时，可能超过背包的承重量。为解决不可行性提出了以下两种方法：

(1) 惩罚法；

(2) 解码法。

惩罚法给每个不可行解指定一个惩罚值。解码法则用贪婪近似法将不可行解转化为可行解。

1. 惩罚法

Gordon 和 Whitley 给每个不可行解一个简单的罚值。该罚值等于背包承重量的超过量[179]。Olsen 对 0-1 背包问题试验过三类惩罚法 [321]。第一种方法的适值函数取为如下乘子形式：

$$eval(\mathbf{x}) = f(\mathbf{x})\ p(\mathbf{x}) \tag{3.4}$$

对于极大化问题惩罚函数构造为

$$p(\mathbf{x}) = 1 - \frac{\left|\sum_{j=1}^{n} w_j x_j - W\right|}{\delta} \tag{3.5}$$

其中

$$\delta = \min \left\{W,\ \left|\sum_{j=1}^{n} w_j - W\right|\right\} \tag{3.6}$$

惩罚项的曲线如图 3.1 所示。

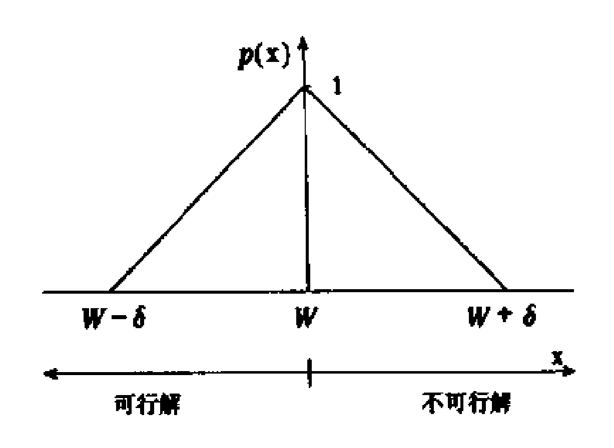


图 3.1 惩罚函数的说明

从图中可见，惩罚函数不仅惩罚超过背包承重量的不可行解，也惩罚背包承重量利用不足的可行解。对于这类最优解位于可行域与不可行域边界上的问题，这种惩罚函数正是所需要的。对于背包问题的线性松弛问题，最优解通常在这个边界上。

2. 解码法

解码法是 Gordon 和 Whitley 提出的[179]。解码过程包括两步：

步骤 1：将 $x_j=1$ 的项目按价值-重量比的降序排列；

步骤 2：按优先适合启发式选择项目，直到背包不能再装为止。

Gordon 和 Whitley 对两个问题试验了上述两种方法：一个 20 个项目的较难问题，一个 80 个项目的较容易问题。对于 20 个项目的问题惩罚法较好，而对于 80 个项目的问题解码法较好。

3.2.2 顺序表达法

该方法是由 Hinterding 提出的[216]。对于一个 n 个项目的问题，染色体由 n 个基因构成，每个基因用一个不同的正整数代表一个项目。项目在顺序中的位置可以看作是该项目选人背包的优先级。按项目的顺序用优先适合启发式方法就可产生一个可行解。

先看表 3.1 中的例子。染色体［1 6 4 7 3 2 5］被解码为可行解［1 6 4 5］在重量限制为 100 的条件下，价值为 73。该方法采用均匀交叉和换位变异作为遗传算子。

表 3.1 7 项目背包问题

项目号	1	2	3	4	5	6	7
重量	40	50	30	10	10	40	30
价值	40	60	10	10	3	20	60

适值函数如下：

$$eval(\mathbf{x}) = \frac{\sum_{j=1}^{n} p_j x_j}{\sum_{j=1}^{n} p_j} \tag{3.7}$$

它给出了一个 0 和 1 之间的适值，通常该值总是小于 1，除非背包能装下所有项目。

这种染色体到可行解的映射是多对一的。这个特点使得该方法大大地减少了种群的多样性。因此，应该用大的种群来增加种群的多样性。

3.2.3 变长表达法

该方法属于直接编码法[216]。染色体中的基因构成一个可行的背包。由于可行背包中的项目数是不固定的，染色体的长度是可变的，而且染色体中项目的顺序没有意义。初始化过程包括如下两步：

初始化过程

步骤 1：产生一个随机的项目顺序；

步骤 2：基于这个顺序按优先适合启发式构造一个可行背包。

插入交叉可以用来处理变长染色体。它是 Falkenauer 和 Delchambre 的分组交叉的简化[123]。其计算步骤如下：

交叉过程

步骤 1：在第一个双亲上选择一个断点，在第二个双亲上选一截片断；

步骤 2：将片断插入第一个双亲的断点处；

步骤 3：删除片断外的重复基因，得到一个原始后代；

步骤 4：按优先适合启发式从原始后代中选择基因，即得到一个可行背包。

以上过程可用图 3.2 来说明。

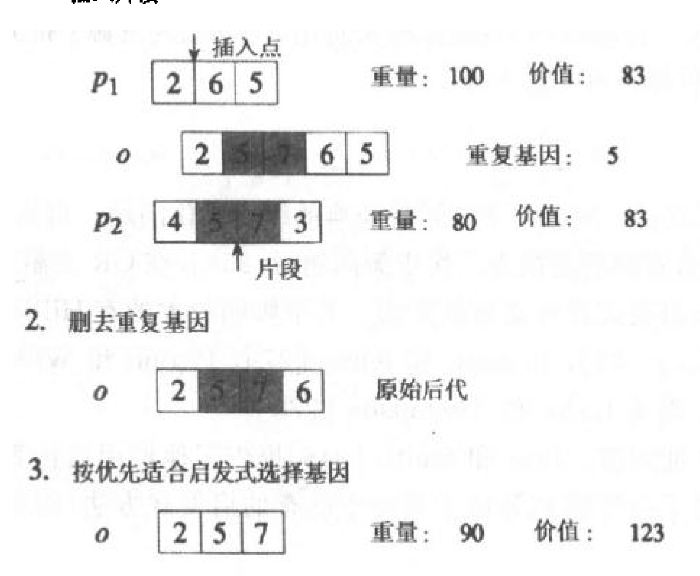


图 3.2 插入交叉的说明

变异与 Falkenauer 和 Delchambre 组变异算子类似，计算步骤如下：

变异过程

步骤 1：随机地删除一个基因；

步骤 2：在顺序中随机地加上一个染色体中没有的基因；

步骤 3：对于以上形成的原始后代用优先适合启发式产生一个可行的背包。

对于这种表达方式，因为染色体只含有选入背包的项目，变异则只能增加一个新项

目，因此探索其他项目的选择组合需要更多的遗传代数。

3.3 二次指派问题

指派问题是一类特殊的线性规划问题，其中工作对资源的需求是一对一的。每样资源(如雇员、机器、时间段)唯一地指派给一件工作(如任务、位置、事件)。资源 $i(i=1,2,\cdots,n)$ 指派给工作 $j(j=1,2,\cdots,n)$ 产生一个相应费用 c_{ij}，问题的目标是如何指派可使总费用达到极小。

用 0-1 变量 x_{ij}表达资源与工作的关系如下：

$$x_{ij}=\begin{cases}1, & \text{资源 } i \text{ 指派给工作 } j\\ 0, & \text{其他}\end{cases}$$

指派问题可用如下数学公式描述：

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{3.8}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij}=1; \quad i=1,2,\cdots,n \tag{3.9}$$

$$\sum_{i=1}^{n} x_{ij}=1; \quad j=1,2,\cdots,n \tag{3.10}$$

$$x_{ij}=0 \text{ 或 } 1; \quad i,j=1,2,\cdots,n \tag{3.11}$$

在指派中引入一种新的费用：指派相关费用。令 a_{ijkl}为资源 i 指派给 j，且资源 k 指派给 l 时的费用，则目标可以表述如下：

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ijkl}x_{ij}x_{kl} \tag{3.12}$$

以上扩展就得到二次指派问题。该问题是经典的组合优化问题。自从 Koopmans 和 Beckmann 首先将设备布置问题建模为二次指派问题后[250]，在 OR 文献里有许多作者提出了一批针对该问题的启发式算法或数值算法。其中影响较大的有 Hillier 和 Connors[213]，Burkard 和 Bonniger [52]，Bazaraa 和 Kirca [27]，Picone 和 Wilhelm [337]，Heragu 和 Kusiak [211]，以及 Kaku 和 Thompson [238]。

该问题是 NP 难问题。Tate 和 Smith [348]报告了他们用遗传算法解二次指派问题的研究结果，证明了遗传算法等价于或好于已有的启发式方法，而且没有过多的辅助计算开销。

3.3.1 编码

Tate 和 Smith 的论文中采用了顺序表达法。对于测试问题，可能的机器位置是平面上的矩形格子，位置之间的距离可用 Euclidean 范数或 Manhattan 范数计算。这些位置按行顺序编号，和可行的机器位置{1,2,…,n}指派一一对应。矩形的行列数是预先指定的。图 3.3 给出了一个 9 个位置的问题。例如染色体[7 3 1 2 9 5 6 8 4]，其中，每个正整数代表对应序号的机器，它在染色体中的序号代表该序号对应的位置。任意机器和位置的交换都得到可行的指派。

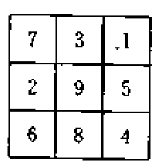


图 3.3　9 位置问题的例子

3.3.2　遗传算子

在 Tate 和 Smith 的论文中提出了如下交叉算子[398]：

交叉过程

步骤 1：任何在双亲中指派到相同位置的机器在后代中仍占据这个位置；

步骤 2：对于剩下的位置由双亲中指派到该位置的两台机器中随机选一台占据，从左到右进行；

步骤 3：将剩下的未指派的机器分派给尚空闲的位置。

以上过程的说明见图 3.4。



图 3.4　交叉运算的说明

变异则随机选择两个位置，并将两点间的子顺序颠倒。这种变异通常称为逆变异。

采用逆变异的遗传算法的基本流程如下：

变异过程

步骤 1：随机产生初始种群，通常都是可行的。

步骤 2：重复以下步骤：

　步骤 2.1：选择双亲；

　步骤 2.2：繁殖后代，并加入种群；

　步骤 2.3：在现行种群中对一些个体进行变异；

　步骤 2.4：组成新的确定数量的种群。

步骤 3：直到给定的停止判据被满足。

广为人知的采用 Euclidean 距离的二次指派问题是 36 个位置，34 个元件的计算机元件安排问题，该问题最早是由 Steinberg 提出的[387]。为把 34 个元件分配到 4×9 个位置上，引入两个空元件。他们在实验中采用了三组繁殖和变异的参数：后代比 25%和变异率 75%，后代比 50%和变异率 50%，后代比 75%和变异率 25%。这里，后代比为产生后代的数量与种群大小的比。变异率为一代中每个解发生变异的概率。遗传算法对三组

参数,10个随机种子进行运行。每次运行中种群由随机产生的100个解构成,计算2 000代。表3.2给出了每组参数的最好目标函数值、用10个随机种子运算的目标值的均值和方差。表3.3给出了找到最好解的最少搜索解数目(记在"速度"一栏),搜索解数的均值和方差。表3.4给出了每组参数10次运行找到的最好解和已知最好解的比。他们还用不同替代策略的遗传算法进行了实验。在100%选择时,每代的染色体都是前代染色体变异的产物。这时,收敛较慢,且不收敛于最好解。而另一个极端是每代只替代种群中最差的25%的个体,使得收敛和采用繁殖和变异算法一样快。

表3.2 解的目标函数值

问题	最好	均值	方差
25%C/75%M	4296.0	4473.1	0.023
50%C/50%M	4271.5	4382.2	0.030
75%C/25%M	4385.6	4490.5	0.016

表3.3 找到最好解前搜索解的数量

问题	速度	均值	方差
25%C/75%M	32125	123650	0.410
50%C/50%M	38059	145674	0.333
75%C/25%M	26093	122331	0.446

表3.4 GA的解和已知最好解的比

问题	最好	最坏
25%C/75%M	1.752	10.602
50%C/50%M	1.173	10.614
75%C/25%M	3.836	8.942

Tate和Smith的实验结果说明他们的遗传算法的性能是非常令人鼓舞的。简单的变异和交叉就能和以前提出的知名的启发式方法媲美,且不必小心地选择变异率、种群大小以及其他参数。

3.4 最小生成树问题

最小生成树问题MST(Minimum Spanning Tree)在组合优化领域里有着悠久的历史。该问题最早是由Boruvka于1926年提出的,他在南Moraria农村电气化中首先研究了MST问题,从而找到了最经济的电网铺设方案[181]。此后,MST的模型被应用到许多组合优化的问题中,如交通问题,通信网设计以及分布式系统等[247]。与此同时,Kruskal[255],Prim[340],Dijkstra[109]和Sollin[383]等提出了几种解MST问题的多项式时间算法。

然而在最近10年里,MST的扩展问题吸引了许多研究者的兴趣,如Narula和Ho的度约束最小生成树(dc-MST)[314],Bertsimas的概率最小生成树(p-MST)[32],Ishii等的随机最小生成树(s-MST)[379],Xu的二次最小生成树(q-MST)[422],Myung,Lee和Tcha广义最小生成树(g-MST)[309],Gilbert的最小Steiner树(MStT)[168],等。

MST的扩展一般是NP难问题,不存在多项式时间算法。由于问题的复杂性,最近一些研究者用遗传算法来解这些问题,包括Hesser等解MST [212],Palmer[328]和Abuali [1]解p-MST,Gen和Zhou解dc-MST [166]。和传统的方法对比,遗传算法取得了满意的结果。使用GA解MST扩展问题的研究工作给组合优化问题带来了新经验。本节将集中讨论用GA解dc-MST问题的方法。

3.4.1 问题的描述

考虑一个连通的无向图 $G = (V,E)$,其中,$V = \{v_1, v_2, \cdots, v_n\}$是代表终端或通信站的有限的端点集。$E = \{e_{ij} | e_{ij} = (v_i, v_j), v_i, v_j \in V\}$是代表终端或站间连线的有限边集。每条边有一个记为 $W = \{w_{ij} | w_{ij} = w(v_i, v_j), w_{ij} > 0, v_i, v_j \in V\}$的正实数的权重,代表距离、价格等。端点和边有时也称为节点和连线。

生成树是连接V中所有端点的来自E的最小边集,因此对于图G至少可找到一棵生成树。最小生成树,记为T^*,是边的权重和最小的生成树。其描述如下:

$$T^* = \min_T \sum_{e_{ij} \in E} w_{ij} \tag{3.13}$$

其中,T为图G的生成树的集合。

端点的度是和该端点相连的边的数量。只有一条边相连的端点称为叶。所以树上的叶的度为1,非叶端点的度大于1。

在一些实际问题中,端点的度不是可任意选取的。比如在道路系统设计中多数情况是四条路构成一个十字路口。通信网络中为防止节点故障带来的脆弱性,对节点的度也有限制。即,确定极小化总的边权的生成树时,往往要求端点v_i的度d_i不大于一个给定值b_i。令T^*_{dc}是图G的所有树中满足度约束的最小生成树,则问题可描述为

$$T^*_{dc} = \min_T \sum_{\substack{e_{ij} \in E \\ d_i \le b_i}} w_{ij} \tag{3.14}$$

为方便起见,这里仅讨论完全图上的对称问题,即,$w_{ij} = w_{ji}$,且$b_i = b$对所有$v_i \in V$。

3.4.2 树的编码

对于MST问题,如何给树编码是GA方法的关键。已有的树的编码方法可分为如下几类:

(1) 边编码;

(2) 端点编码;

(3) 端点和边混合编码。

不论什么编码,有效的编码应具有如下特点[329]:

(1) 能够表达所有可能的树且表达的都是树,没有树不能被表达;

(2) 所有树都被平等无偏的表达,即所有树应表达为相同的编码数;

(3) 编码和树之间可以方便地来回转换,并可容易地计算适值函数和约束;

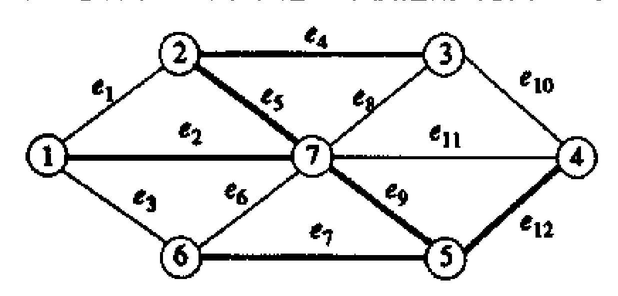
(4) 能够用较短的程序即可使种群选入更适合的染色体;

(5) 表达上的小变化能够引起树的小变化。

按上述准则我们说明以上三类编码方法。

1. 边编码

对每条边指定下标 k, 即 $E=\{e_k\}$, $k=1,2,\cdots,K$, 其中 K 是图中的边数。一个指明边是否在生成树上的二进制串可用来表达一个候选解,见图 3.5。



e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}
0	1	0	1	1	0	1	0	1	0	0	1

图 3.5 图及其生成树的边的编码

这种边编码实际上是一种树的直观表达方法。然而,对于一个 n 端点的图,一棵树应是一个连接 $n-1$ 条边的无回路的子图。但上述边编码不能表达这种性质。若二进制串含有非 $n-1$ 条边,则不是一棵树。即使正好含有 $n-1$ 条边,也很可能不是树,因为其中可能有回路。的确,随机的二进制编码获得生成树的概率随端点数 n 的增加而趋于无穷小。在初始种群中或交叉与变异运算之后种群中可能没有一棵生成树。

Piggott 和 Suraweera 对 MST 问题用过这种边编码,它从正确的边数开始,并在遗传运算中保持这个边数[338]。然而,总的说来,这种编码方法对于 MST 问题不大适用,因为它获得树的概率极低。

2. 端点编码

图论计算中的一个经典的定理是 Cayley 定理,即在一个 n 个端点的完全图中有 $n^{(n-2)}$ 个不同的树。Prüfer 对基于这类树和 $n-2$ 个数字的所有排列的集合的一一对应关系给出了 Cayley 定理的结构化证明[342]。这表明可以仅用 $n-2$ 数字的排列来唯一地表达一棵树,其中每个数字都是 1 和 n 之间的整数。这个排列通常称为 Prüfer 数。

使用 Prüfer 数编码树是端点编码法中的一种。任意树都至少有两个叶端点[381]。基于这个现象,可以容易地构造如下编码方法:

编码过程

步骤 1:令端点 i 是标定树 T 上的标号最小的叶子。

步骤 2：令唯一与 i 相连的端点 j 作为编码的第一个数字。这里，编码的顺序是从左读到右。

步骤 3：删去端点 i 及 i 到 j 的边，于是得到一棵 $n-1$ 个端点的树。

步骤 4：重复上述运算，直到只剩下一条边。于是得到了一个 Prüfer 数或一个 $n-2$ 个 1 和 n 之间的数字的排列。

按如下算法，由 Prüfer 数可以唯一地产生一棵树：

解码过程

步骤 1：令 P 是原 Prüfer 数，$\overline{P}$ 是所有不包括在 P 的端点的集合，$\overline{P}$ 即为构造一棵树的合格端点。

步骤 2：令 i 为标号最小的合格端点。令 j 为 P 的最左边的数字。将 i 到 j 的边加到树上。标号 i 不再是合格的，将其从 $\overline{P}$ 除去，并从 P 中除去 j。若 j 在 P 的剩余部分中不再出现，将 j 加入 $\overline{P}$ 中。重复以上过程直到 P 中不再有数字为止。

步骤 3：若 P 中不再有数字，$\overline{P}$ 中正好有两个合格端点，r 和 s，将 r 到 s 的边加到树上，于是构成了一棵 $n-1$ 条边的树。

下面的例子是这种编码方法的说明。Prüfer 数（3 3 1 1）表示了一棵 6 个端点的完全图上的生成树，如图 3.6 所示。Prüfer 数的构成过程如下：找出具有最小标号的叶端点。对于本例即为端点 2。由于端点 3 是树上唯一与端点 2 相邻的端点，指定 3 为 Prüfer 数的第一个数字。删去 2 和边(2,3)。对剩余的子树重复以上过程，直到仅剩下(1,6)为止。于是得到了这棵树的 4 个数字的 Prüfer 数。

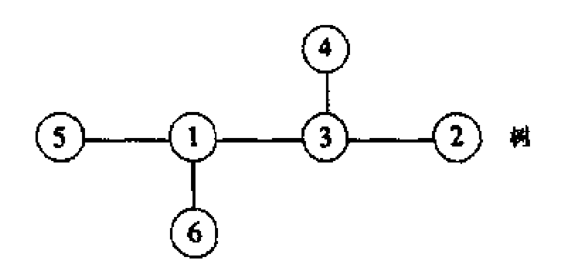


图 3.6　树及其 Prüfer 数

反过来，对于 Prüfer 数 $P=$ (3 3 1 1)，端点 2，4，5 和 6 是合格端点，即 $\overline{P}=$ [2,4,5,6]。端点 2 是标号最小的合格端点，端点 3 是 P 中最左边的数字。将边(2,3)加到树上，从 $\overline{P}$ 中除去端点 2，并除去 P 中左边的数字 3，令 $P=$ (3 1 1)。端点 4 是现在标号最小的合格端点，第二个端点 3 是剩下的 P 中最左边的数字。然后将边(4,3)加到树上，从 $\overline{P}$ 中除去端点 4，并从 P 中除去数字 3，得到 $P=$ (1 1)。由于端点 3 已不再在剩下的 P 中，端点 3 成为合格端点，并将其加到 $\overline{P}$ 中，得到 $\overline{P}=$ [3,5,6]。这时端点 3 是标号最小的合格端点，将边 (3,1) 加到树上，除去 P 左边的数字 1，并从 $\overline{P}$ 除去端点 3，再作下一步运算。现在 $P=$ (1) 且只有端点 5 和 6 是合格的。将边 (5,1) 加到树上，除去 P 中的最后一个数字，且端点 5 不再是合格的，将其从 $\overline{P}$ 中除去。由于端点 1 不再在 P 中、成为合格的，将其加入 $\overline{P}$，得到 $\overline{P}=$ [1,6]。P 已是空的，只有端点 1 和 6 是合

格的。将边 (1,6) 加到树上,停机。于是得到了图 3.6 所示的树。

Prüfer 数是给生成树编码的最合适的方法,特别是对 MST 的扩展问题,比如 dc-MST。这种编码方法占存储较小,正如 Palmer 和 Kershenbaum 指出的,即使 Prüfer 数的一个数字变化,也会带来树的意想不到的变化[329]。例如,Prüfer 数 3341 和 3342,只有一个数字不同,但它们表达的五条边的树中,只有两条是相同的,如图 3.7 所示。

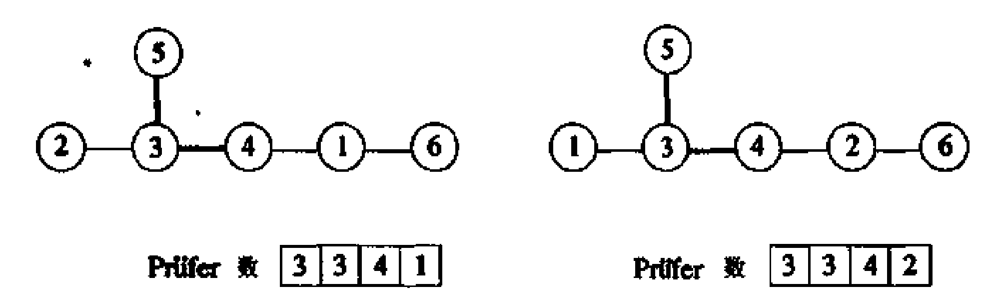


图 3.7 Prüfer 数 3341 和 3342 的说明

Palmer [329]和 Abuali[1]还介绍了一些其他的端点编码方法,只是这些编码需要复杂的算法来保持树不被破坏。

3. 边和端点混合编码

边和端点混合编码又称为连线节点倾向编码,是由 Abuali 等[1]和 Palmer 等[328]提出并发展起来的。按这种编码,染色体中每个节点和连线都有一个倾向值。节点和连线的倾向值是 0 到 255 之间的整数。基于这种编码,运用 Prim 的最小生成树算法,改进费用矩阵 $C'=[c'_{ij}]$,就可获得最小生成树。其中

$$c'_{ij} = c_{ij} + p_1 b_{ij} c_{max} + p_2(b_i + b_j) c_{max} \tag{3.15}$$

c_{max}是图的最大连接费用;b_{ij}是节点 i 与节点 j 之间的连接倾向;b_i 是节点 i 的连接倾向;p_1 和 p_2 是控制参数。Palmer 在实验中取 $p_1=0$ 和 $p_2=1$。

实际上,这种编码并不直接构成树,只是一个修正费用矩阵。基于修正费用矩阵,可用 Prim 算法来产生树[340]。这种编码方法有三个显著的缺点:

(1) 编码较长(存储费用);

(2) 需要一个最小树生成算法由编码产生树(计算费用);

(3) 缺乏一些和树有关的信息,如度、连接关系等。

所以很难将其用于 dc-MST 问题。

3.4.3 遗传算法方法

1. 染色体表达

dc-MST 问题是 MST 问题加上度约束的扩展,所以染色体表达中应含有显式的或隐式的各个端点的度的信息。在几种树的编码中,只有 Prüfer 数编码含有显式端点的度的信息,在 Prüfer 数编码中度为 d 的端点正好出现 $d-1$ 次。因此,通常采用 Prüfer 数编码。

2. 交叉和变异

Prüfer 数编码任意交叉和变异后仍代表一棵树。简单地采用单点交叉的说明见图 3.8。变异则可随机地变为 1 和 n 中的一个整数，见图 3.9。

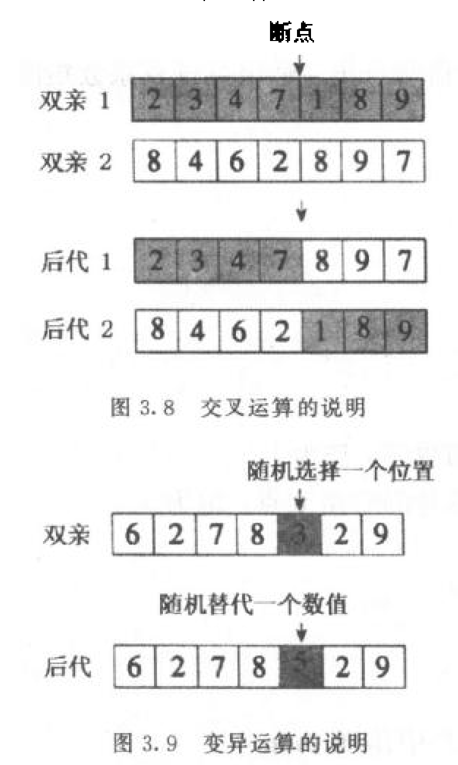


图 3.8 交叉运算的说明

图 3.9 变异运算的说明

3. 度的改进

由于存在度约束，无论是初始种群中随机产生的染色体，还是交叉或变异产生的后代都可能不满足度约束，即为非法的。因此需要改进染色体中的非法端点，而采用 Prüfer 数编码则容易做到。

令 $\overline{V}_{dc}$是染色体中未作度检查的端点集。如一个端点违反度约束，即编码中该端点出现的次数大于 $d-1$，那么将多出的端点随机地替换为 $\overline{V}_{dc}$的其他端点就可减少该端点的度。例如，在一个端点度不超过 3 的 9 端点完全图中，染色体(2 6 6 2 6 8 1)中，端点 6 违背约束，因为它出现了 3 次。改进过程如图 3.10 所示。



图 3.10 带度约束染色体的改进

4. 评估与选择

评估过程由两步组成：

(1) 转换染色体为树；

(2) 计算树的总权。

令 P 为染色体，$\overline{P}$ 是合格端点集。适值可按权系数矩阵 $W=[w_{ij}]$计算。评估过程说明如下：

评估过程

begin

　$T \leftarrow \{\phi\}$；

　$eval\ (T) \leftarrow 0$；

　按照 P 定义 $\overline{P}$；

　repeat

　　选择 P 中最左边的数字，记为 i；

　　从 $\overline{P}$ 中选择最小标号的合格端点，记为 j；

　　$T \leftarrow T \cup \{e_{ij}\}$；

　　$eval\ (T) \leftarrow eval\ (T) + w_{ij}$；

　　从 P 中删除 i；

　　从 $\overline{P}$ 中删除 j；

　　if i 不再在剩余的 P 中出现 **then**

　　　将 i 加入 $\overline{P}$；

　　end

　　$k \leftarrow k+1$；

　直到 $k \leq n-2$

　$T \leftarrow T \cup \{e_{rs}\}$，　r、$s \in \overline{P}$；

　$eval\ (T) \leftarrow eval\ (T) + w_{rs}$；

end

选择过程可采用 $(\mu+\lambda)$ 选择和转轮选择的混合。$(\mu+\lambda)$ 选择是 Bäck 和 Hoffmeister 提出的[13]，它从 μ 个双亲和 λ 个后代中选择 μ 个最好的染色体。如没有 μ 个不同的染色体，种群中的空缺则用转轮选择来填满。具体做法如下：

选择过程

begin

　选择 μ' 个不同的染色体；

　if $\mu' < \mu$ **then**

　　用转轮法选择 $\mu - \mu'$ 个染色体；

　end

end

5. dc-MST 算法

和传统的遗传算法相比，该算法的主要特点是为满足度约束在简单遗传算法中嵌入了度改进步骤。整个 dc-MST 计算步骤如下：

dc-MST 遗传算法步骤

```
begin
t ←0;
  初始化 P(t);
  改进 P(t)的度;
  评估 P(t);
  While 不满足终止条件 do
    重组 P(t)获得 C(t);
    改进 C(t)的度;
    评估 C(t);
    从 P(t)和 C(t)中选择 P(t+1);
    t ← t + 1;
  end
 end
```

6. 数值例子

以下例子是 Savelsbergh 和 Volgenant 给出的，它们用分枝定界算法求得的最优解的值为 2256 [368]。这是一个 9 节点的完全图，见表 3.5。

表 3.5　9 节点 dc-MST 问题的边的权重

i	1	2	3	4	5	6	7	8	9
1	0	224	224	361	671	300	539	800	943
2	224	0	200	200	447	283	400	728	762
3	224	200	0	400	566	447	600	922	949
4	361	200	400	0	400	200	200	539	583
5	671	447	566	400	0	600	447	781	510
6	300	283	447	200	600	0	283	500	707
7	539	400	600	200	447	283	0	361	424
8	800	728	922	539	781	500	361	0	500
9	943	762	949	583	510	707	424	500	0

遗传算法的参数设定如下：种群大小 $pop_size = 50$，交叉率 $p_c = 0.5$，变异率 $p_m = 0.01$，最大代数 $max_gen = 500$，所有端点的限定度 $b = 3$，运行 30 次。

用该 GA 算法，多数运行都能达到最优值 2 256。图 3.11 清楚地表明，GA 法不用启发式或问题的专门性质，就能通过表达解的染色体的繁殖过程以高的概率达到最优解。

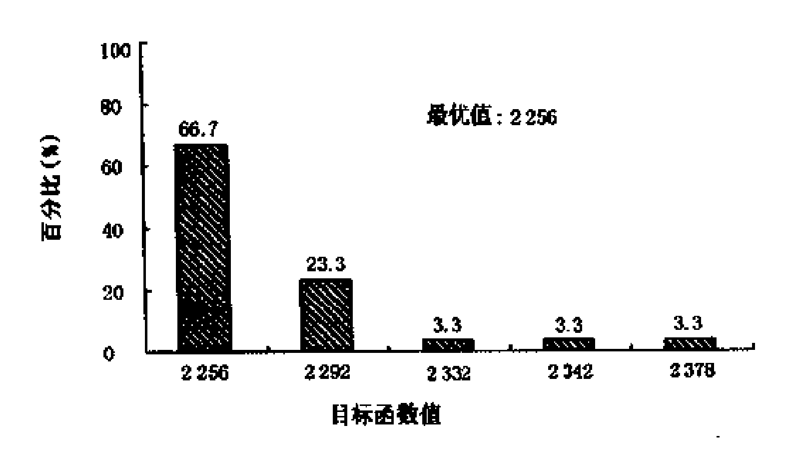


图 3.11　遗传算法解 dc-MST 问题的解的分布

我们比较了两种选择策略。表 3.6 说明两种方法都能获得最优解，但采用混合选择策略的 GA 比转轮法的平均目标值和相对误差都要好一些。

表 3.6　两种选择策略的结果比较

方　　法	最优值	平均目标值	相对误差(%)
采用混合选择策略的遗传算法	2 256	2 270.7	0.6
采用转轮选择策略的遗传算法	2 256	2 365.2	4.8

图 3.12 说明采用混合选择策略的遗传算法比采用转轮选择的遗传算法具有更好的寻优机理。前者在 30 次运行中 20 次达到最优解，而后者在 30 次运行中只有 2 次达到最优解。这些结果表明混合选择策略对于组合优化问题比旋轮选择方法更合适。

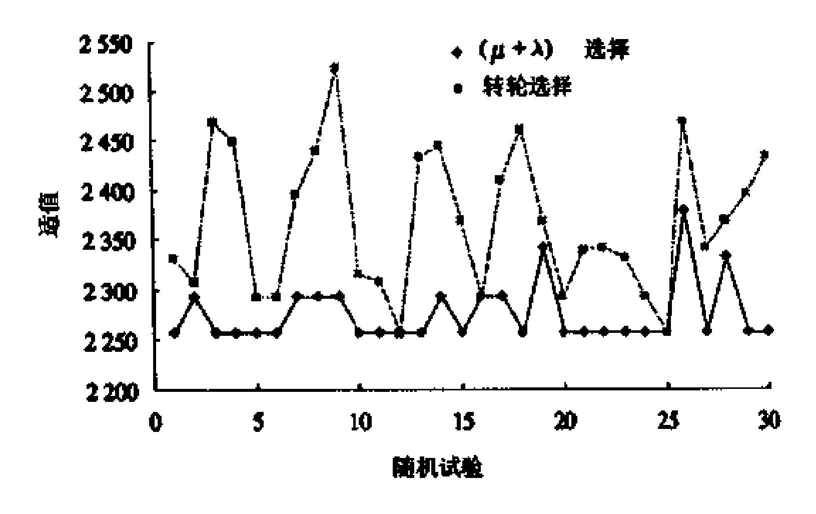


图 3.12　两种选择策略的遗传算法比较

3.5　旅行商问题

旅行商问题 (TSP) 是组合优化中研究最多的问题之一。问题的叙述极为简单：一个商人要找一条通过 n 个城市的最短巡回。从 50 年代中起，出版了大量关于本问题的文献。Lawler 编辑的书对到当时为止的所有主要研究成果作了全面深入的综述[263]。在过去

的20年里，在求旅行商问题的最优解方面取得了极大的进展。主要的成就包括48城市问题[90]，120城市问题[193]，318城市问题[87]，532城市问题[326]，666城市问题[194]，2 392城市问题[327]。Reinelt在他的专题文章中指出，尽管有这些成就，但旅行商问题还远未解决。问题的许多方面还要研究，很多问题还在期待满意的回答[352]。

旅行商问题吸引了许多不同领域的研究者，包括数学、运筹学、物理、生物和人工智能等领域。这是因为TSP展示了组合优化的所有方面，它已经成为并将继续成为测试新算法的标准问题，如模拟退火、禁忌搜索、神经网络以及进化方法等都用TSP来测试。较早用遗传算法来解TSP的论文有Goldberg和Lingle [174]，Grefenstette [185]。此后，TSP就成了遗传算法学界的一个主要研究目标[287]。对GA/TSP的研究为遗传算法解组合优化问题提供了丰富的经验和牢固的基础。粗略地说，主要的努力在以下三个方面：

(1) 采用适当的表达方法来给巡回编码；

(2) 设计可用的遗传算子，以保持父辈特性并避免不可行性；

(3) 防止过早收敛。

对于前两个方面，我们将在下一节中详细介绍。而对第三个方面，将参考Tsujimura，Gen和Cheng工作，他们利用跳出局优的信息熵的概念介绍了一种多样性的测量方法[404]。

3.5.1 表达

染色体通常采用简单的二进制串，但这种简单的表达方法不能较好地适用于TSP和其他组合优化问题。过去10年里，为TSP提出了几种染色体表达方法，其中，换位表达和随机键表达不仅适用于TSP，也适用于其他组合优化问题。

1. 换位表达

这种表达大概是TSP巡回的最自然的表达，其中城市是按访问的顺序排列的[101，287]。这种表达的搜索空间是城市顺序换位的集合。例如，对于一个9城市的TSP：

$$3-2-5-4-7-1-6-9-8$$

可简单表示为

$$[\,3\ 2\ 5\ 4\ 7\ 1\ 6\ 9\ 8\,]$$

以上表达也称为路径表达或顺序表达。这种表达采用传统的单点交叉时，可能导致非法的巡回。为此，人们研究了多种交叉算子。

2. 随机键表达

随机键表达首先是由Bean提出的[29]。这种表达方法将解表达为一串(0，1)间的随机数。这些随机数用作解码的分类键。例如，一个9城市问题的一个染色体为

$$[\,0.23,\ 0.82,\ 0.45,\ 0.74,\ 0.87,\ 0.11,\ 0.56,\ 0.69,\ 0.78\,]$$

其中，编码中位置i代表城市i，位置i的随机数表示城市i在TSP巡回中的顺序。我们将这些随机键按升序排列，得到如下巡回：

$$6-1-3-7-8-4-9-2-5$$

随机键消除了解的表达中的不可行性。这种表达方法可用于各种顺序优化问题，包括机器

调度、资源分配、车辆线路和二次指派问题等。

3.5.2 交叉算子

过去10年里，为换位表达设计了好几种交叉算子，如部分映射交叉(PMX)、顺序交叉(OX)、循环交叉(CX)、基于位置的交叉、基于顺序的交叉、启发式交叉等。

这些算子大略可分为两类：

(1) 规范法；

(2) 启发式法。

规范法可看作二进制串的两点或多点交叉在换位表达中的扩展。一般，换位表达在做简单的两点或多点交叉时，后代中一些城市会失去，一些城市会重复，从而得到非法染色体。为此，需要嵌入修复程序来解决后代的非法性。

规范法的修复程序是盲目的，它不能保证交叉后的后代比双亲更好。而启发式法的目的是为了产生改进的后代。

1. 部分映射交叉(PMX)

PMX是由Goldberg和Lingle提出的[174]。PMX可看作二进制串的两点或多点交叉在换位表达中的扩展，它用特别的修复程序来解决简单两点交叉引起的非法性。所以PMX基本上是简单的两点交叉加修复程序。其步骤如下：

PMX 步骤

步骤1：在字串上均匀随机地选择两点，由这两点定义的子串称为映射段；

步骤2：交换双亲的两个子串，产生原始后代；

步骤3：确定两映射段之间的映射关系；

步骤4：根据映射关系将后代合法化。

这个步骤可用图3.13来说明。在原始后代中，城市1，2和9重复，而城市3，4和5却丢失了。按步骤3确定的映射关系，重复的城市1，2和9将分别被丢失的城市3，5和4替代，但交换的子串却保持不变。

2. 顺序交叉(OX)

OX是由Davis提出的[99]。它可看作是一种带有不同修复程序的PMX的变型。OX步骤如下：

OX 步骤

步骤1：从第一双亲中随机选一个子串；

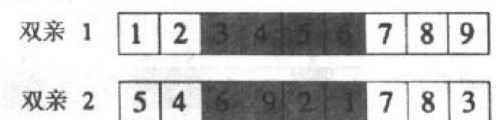
步骤2：将子串复制到一个空字串的相应位置，产生一个原始后代；

步骤3：删去第二双亲中子串已有的城市，得到原始后代需要的其他城市的顺序；

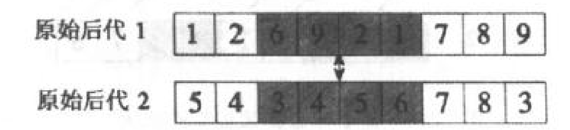
步骤4：按照这个城市顺序，从左到右将这些城市定位到后代的空缺位置上。

这个过程的说明见图3.14。图中给出了产生一个后代的例子。同样步骤可以由同一对双亲产生另一个后代[2 5 4 9 1 3 6 7 8]。

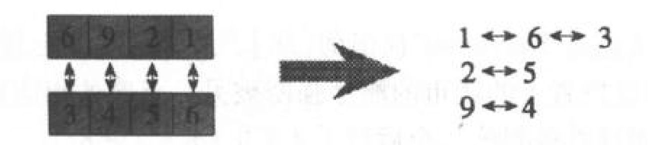
1. 随机选择子巡回



2. 交换双亲中的子串



3. 确定映射关系



4. 用映射关系将后代合法化

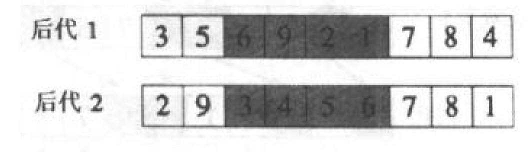


图 3.13 PMX 运算的说明

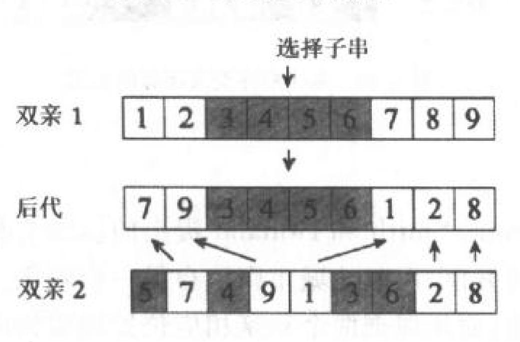


图 3.14 OX 运算的说明

3. 基于位置的交叉

基于位置的交叉是由 Syswerda 提出的[99]。它基本上是一种换位表达的均匀交叉加一个修复程序。它也可以看作是 OX 的一种变型，其中城市的子串是不连续的。基于位置的交叉步骤如下：

基于位置的交叉步骤

步骤 1：从一个双亲上随机地选一组位置；

步骤 2：将这些位置复制到一个空字串的相应位置，产生一个原始后代；

步骤 3：删去第二双亲上该组中已有的城市，剩下的城市构成了原始后代需要的顺序；

步骤 4：按照这个城市顺序，从左到右将这些城市定位到后代的空缺的位置上。

该步骤可用图 3.15 来说明。用同样的步骤可产生另一个后代[2 4 3 5 1 9 6 7 8]。

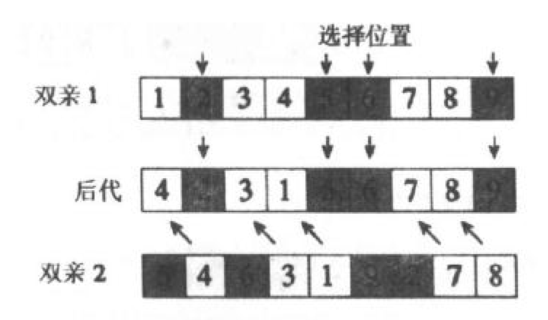


图 3.15　基于位置交叉运算的说明

4. 基于顺序的交叉

基于顺序的交叉也是 Syswerda 提出的[99]。该方法实际上是基于位置的交叉的变型，其中一个双亲选定位置上的城市的顺序强制被另一双亲的相应城市所替代。说明见图 3.16。用同样的步骤可以得到第二个后代 [4 2 3 1 5 6 7 9 8]。

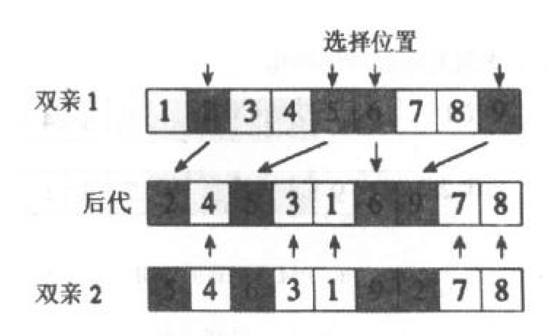


图 3.16　基于顺序交叉运算的说明

5. 循环交叉 (CX)

循环交叉法是由 Oliver，Smith 和 Holland 提出的[320]。和基于位置的交叉一样，该方法从一个双亲中取一些城市，而其他城市则取自另一个双亲。不同之处是来自第一个双亲的城市不是随机产生的，而是根据两个双亲相应位置城市构成的环确定的。CX 的步骤如下：

CX 步骤

步骤 1：根据双亲相应的城市位置找出一个循环；

步骤 2：把一个双亲的循环上的城市复制到一个后代上；

步骤 3：删去另一个双亲的已在循环上的城市，剩下的城市即可用来确定剩余城市的位置；

步骤 4：用这些剩余城市填满后代剩余的位置。

以上过程可用图 3.17 来说明。用同样的方法我们可得到另一个后代[5 4 3 9 2 6 7 8 1]。

6. 子巡回交换交叉

该交叉算子是由 Yamamura，Ono 和 Kobayashi 提出的[425，461]。他们首先从双亲中选择子巡回。子巡回中含有共同的城市，后代则由图 3.18 所示的交换子巡回的方法产生。

1. 找出双亲确定的循环

双亲 1	1	2	3	4	5	6	7	8	9
双亲 2	5	4	6	9	2	3	7	8	1

循环 1 → 5 → 2 → 4 → 9 → 1

2. 将循环中的城市复制给后代

原始后代	1	2		4	5				9

3. 为后代确定剩余城市

双亲 2	5	4	6	9	2	3	7	8	1

剩余城市	6	3	7	8

4. 填满后代

后代	1	2	6	4	5	3	7	8	9

图 3.17 CX 运算的说明

选择子巡回

双亲 1	1	2	3	4	5	6	7	8	9
双亲 2	3	5	6	4	7	9	2	1	8

↓

交换子巡回

后代 1	1	2	3	5	6	4	7	8	9
后代 2	3	4	5	6	7	9	2	1	8

图 3.18 子巡回交换交叉运算的说明

7. 启发式交叉

启发式交叉是由 Grefenstette，Gopal，Rosmaita 和 Gucht 首先提出的[185]。传统的 TSP 问题的启发式算法有两种：最近邻居法和最好插入法。Grefenstette 的交叉法采用的是最近邻居法的原理。Cheng 和 Gen 在车辆路线和调度问题的遗传算法中采用了最好插入法[65,69]。启发式交叉的步骤如下：

启发式交叉步骤

步骤 1：从一对双亲中随机地选一个城市来开始；

步骤 2：选择由现行城市出发的不构成循环的最短边(由双亲表达的)。若两条边构成循环，则随机地选择一个能使巡回继续的城市；

步骤 3：如巡回完成，停机；否则转步骤 2。

Liepins，Hilliard，Palmer 和 Morrow 稍微改进了 Grefenstette 的算法，他们令任意巡回都由同一城市开始[273]。Cheng 和 Gen 也提出了一种改进的启发式交叉法[61]。一

种子巡回旋转程序引入到启发式算法中，以便尽可能多地在后代中保存父辈中好的子巡回。他们针对 Nemhanser 和 Wolsey 的 67 城市问题，对 PMX，CX 及他们的改进算法作了比较[316]。图 3.19 说明了启发式交叉法明显优于其他方法。

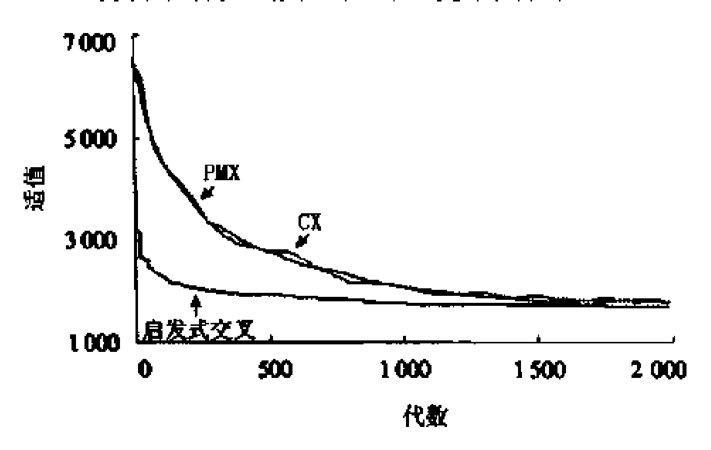


图 3.19 PMX，CX 和启发式交叉的比较

3.5.3 变异算子

对于换位表达变异运算相对比较容易。过去 10 年里，提出了几种用于换位表达的变异算运，如反转、插入、移位以及互换等变异方法。

1. 反转变异

反转变异是在染色体上随机地选择两点，将两点间的子串反转。其说明见图 3.20。

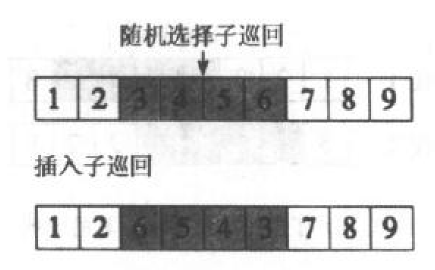


图 3.20 反转变异的说明

2. 插入变异

插入变异是随机地选择一个城市，并将它插入到一个随机的位置中。如图 3.21 所示。

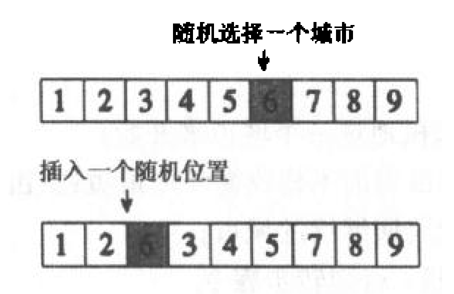


图 3.21 插入变异的说明

3. 移位变异

移位变异是随机地选择一个子巡回，并将其插入到一个随机的位置中，如图 3.22 所示。插入变异可以看作是子巡回只含有一个城市的移位变异。

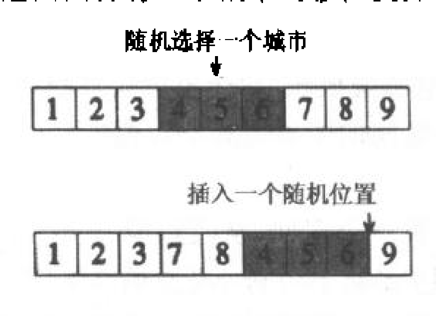


图 3.22 移位变异的说明

4. 互换变异

互换变异是随机地选择两个位置，并将这两个位置上的城市相互交换，如图 3.23 所示。这种变异实质上是 TSP 问题的 2 优启发式。

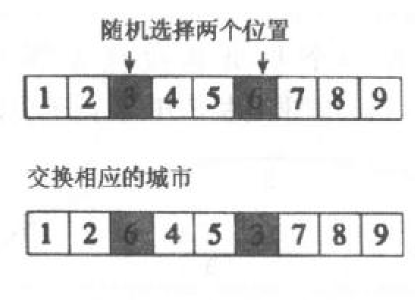


图 3.23 互换变异的说明

5. 启发式变异

启发式变异是由 Cheng 和 Gen 提出的[62，64]。这种变异采用了邻域技术，以获得后代的改进。对于一个染色体按它的邻域交换不多于 λ 个基因，可获得一族染色体，选择其中最好的一个作为变异产生的后代。启发式变异过程如下：

启发式变异过程

步骤 1：随机地选出 λ 个基因；

步骤 2：按所有选出基因的可能的换位产生邻域；

步骤 3：评估所有邻域点，选出最好的作为变异产生的后代。

以上过程如图 3.24 所示。

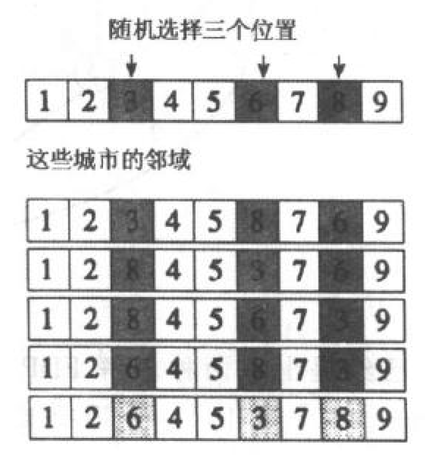


图 3.24 启发式变异的说明

3.6 影片递送问题

影片递送问题(简称 FDP)是组合优化的一个新问题,是由 Cheng 和 Gen 首先提出的[60]。FDP 可以描述如下:

一盘影片拷贝要在 n 个电影院放映。每个电影院放映的次数已定,记为一个非负的整数 $d_i(i=1,2,\cdots,n)$。两个影院间的距离记为 w_{ij},若影院 i 和 j 不能直接相连,则令 $w_{ij}=+\infty$。

问题是要为影片递送员找一个巡回,从主影院 1 开始,将影片拷贝送到第 i 家影院 $d_i(i=1,2,\cdots,n)$ 次,最后回到主影院 1,并极小化总的路线长度。当所有的 $d_i(i=1,2,\cdots,n)$ 为 1 时,FDP 变为经典的 TSP。可见,FDP 是 TSP 的新扩展,它可以推广到一大类路径与排序问题中。Zhang 和 Zheng 研究了 FDP 和 TSP 的转换,并用启发式方法求解了 TSP 的转换问题[430]。

FDP 可以方便地用图来表达。对于付权无向图 $G=(V,E)$,其中 $V=\{v_1,v_2,\cdots,v_n\}$ 代表电影院的有限顶点集,$E=\{e_{ij}|e_{ij}=(v_i,v_j),v_i,v_j\in V\}$ 代表影院间距离的有限的边集。每条边有一个代表距离的非负权重 $W=\{w_{ij}|w_{ij}=w(v_i,v_j),w_{ij}>0,v_i,v_j\in V\}$,每个顶点有一个非负整数代表影片放映的次数,记为 $D=\{d_i|\ d_i\text{非负整数},i=1,2,\cdots,n\}$。用图表达 FDP 的例子见图 3.25。

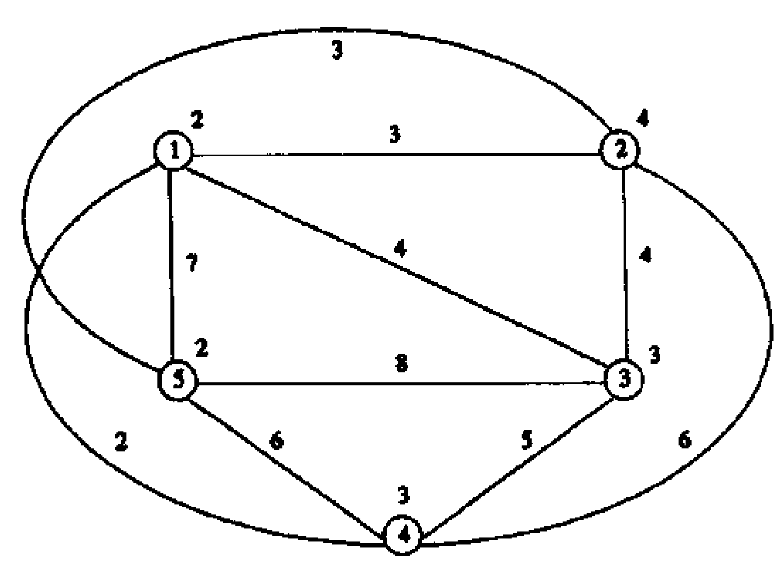


图 3.25 FDP 的图表达

已经提出有两种求解 FDP 的遗传算法,分别由 Cheng 和 Gen [66],Zhang 和 Zheng [431] 提出。一般,FDP 比 TSP 要难解得多。

3.6.1 表达

可行的递送巡回是图上的一个行走(walk)。记住,所谓图 G 上的行走是如下所示的一个有限顶点和边交替的顺序[56]:

$$v_{i_1},e_{i_1i_2},v_{i_2},e_{i_2i_3},\cdots,v_{i_{k-1}},e_{i_{k-1}i_k},v_{i_k}$$

行走中存在着顶点和边的重复。一个顶点 v 的重复构成了 v-v 平凡行走(trivial walk)。一个可行的递送巡回是给定图上的每个顶点不含平凡行走且准确重复 $d_i(i=1,2,\cdots,n)$次的行走。

通常行走只用顶点来标记,因为有了顶点的顺序,边就清楚了。于是,行走可表达如下:

$$v_{i_1},v_{i_2},\cdots,v_{i_n}$$

这即为 FDP 自然的编码方案。先看表 3.7 中的例子。

表 3.7 FDP 的例子

电影院	1	2	3	4	5
放映次数	2	4	3	3	2

一个可行递送巡回(或染色体)表达如下:

[1 2 3 2 5 3 4 2 3 4 2 5 4 1]

它可解释为影片先给影院 1,然后给影院 2,……按此顺序继续。对于 n 个电影院,影院 i 放映 d_i 次的问题,一个合法染色体必须有 i 的 d_i 个重复,因此其长度为

$$l=\sum_{i=1}^{n}d_i \tag{3.16}$$

3.6.2 遗传运算

1. 交叉运算

子巡回保留交叉就是为以上表达方法设计的,因为子巡回是巡回的最自然的组成部分。在巡回中,子巡回以同一个城市的开始和结束来识别。该交叉方法由五步构成:

交叉过程

步骤 1:从一个双亲中随机地选一个子巡回,再从另一个双亲中选一个以同一基因起始终止的子巡回;

步骤 2:交换这对子巡回产生的后代。然而,后代的长度可能非法,即它们可能长于或短于合法染色体,因为两个子巡回通常有不同的长度(基因数不同);

步骤 3:比较子巡回,确定后代中丢失或多余的基因;

步骤 4:删去多余的基因。注意,删除中可能引起位置非法,即有些影院可能与本身相邻;

步骤 5:将丢失的基因插入到非法相邻的位置中,使之合法化。如没有这种位置,则插入到一个随机的位置中。

例如,先从双亲 p_1 中随机地选出一个子巡回 $subtour_1$:[2 3 5 3 4 2],再从 p_2 中选出同样以影院 2 起止的子巡回 $subtour_2$:[2 4 3 1 2]。将 p_1 中的 $subtour_1$ 替换为 $subtour_2$,产生一个后代。显然它是非法的,因为影院 5 和 3 丢失一次,而影院 1 则多余一次。删除多余基因并插入丢失基因使之合法化。注意,删除中造成位置非法,即影院 4 与本身相邻。因此首先将影院 5 插入到该位置中。由于已没有非法位置,影院 3 被插入到一个随机合法位置中。见图 3.26。

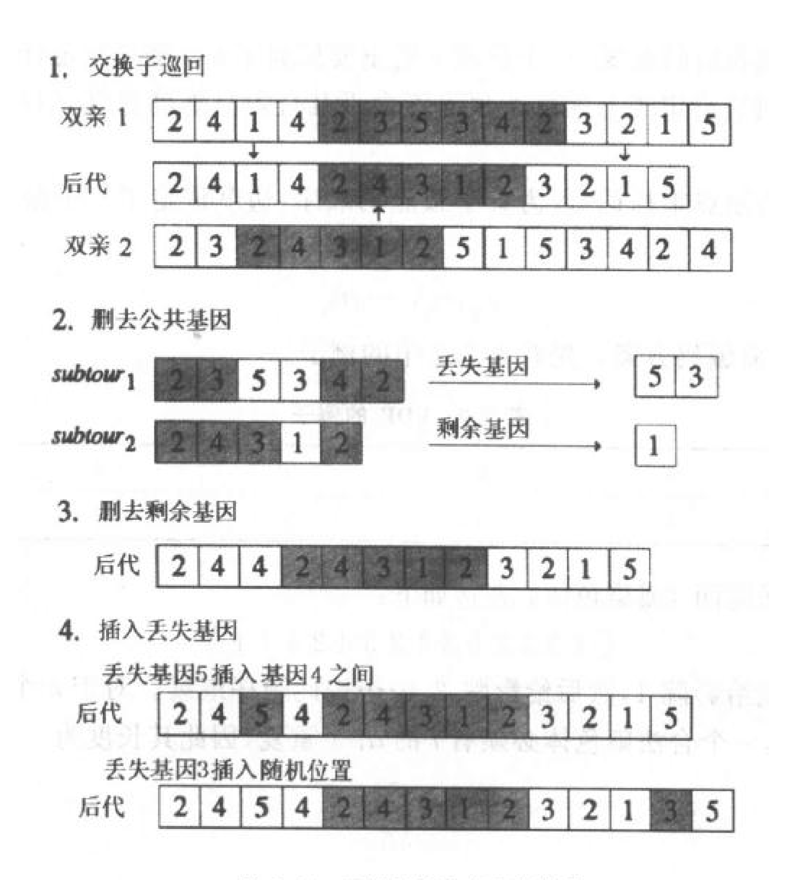


图 3.26 子巡回保留交叉的说明

2. 变异

变异交叉是按邻域搜索机理设计的，以便产生改进的后代。图 3.27 给出了一个例子，其中子巡回的长度 $\lambda = 4$ 是随机选定的。由子巡回中的影院和子巡回外的影院的合法交换（没有影院与本身相邻）可以构成染色体的邻域。

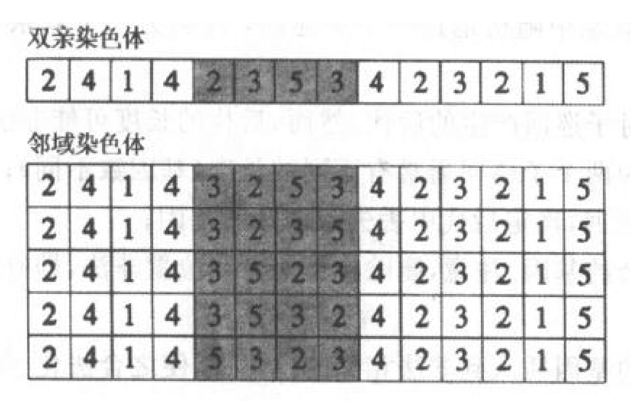


图 3.27 变异的说明

以上方法曾对随机产生的 30 个影院 124 次放映的问题，就不同的参数设定，进行过反复计算，以研究参数设定对遗传算法的影响。结果表明，变异起了关键的作用，且变异概率对遗传算法的性能有明显的影响。

第四章　可靠性优化问题

4.1 引　　言

从广义上讲，可靠性是一个系统性能的度量。随着系统复杂性程度的增加，系统的非可靠性以费用，费力程度和寿命等方面的问题表现出来。因而对系统可靠性的评价，产品和系统可靠性的改善就显得越来越重要。

系统的可靠性可以定义为系统在给定的时间区段和状态下正常工作的概率。在过去的几十年内，可靠性的研究涉及很宽广的领域，包括可靠性分析、失效建模、可靠性优化、可靠性增长及建模、可靠性测试、可靠性数据分析、加速测试及生命周期费用等[346]。

研究可靠性问题的主要目标之一是寻求最好的方法提高系统的可靠性，可靠性优化有助于可靠性工程师实现这一目标。多数的可靠性优化方法均假定系统在串并联中均有冗余的元件，并有替换的设计方案。优化主要集中于冗余元件的最优分配和比较设计的最优选择以满足系统的需求。在过去的 20 年里，一些可靠性优化技术相继问世。一般地，这些技术分为线性规划、动态规划、整数规划、几何规划、启发式方法、拉格朗日乘子法等[345]。有关系统可靠性优化技术的综述文章可参见 Tillman，Hwang，Kuo[402]。

4.1.1　系统可靠性的组合特性

复杂系统常常分解为由单元、子系统或用于可靠性分析的元件组成的功能实体。我们通常用可靠性分析的网络建模技术和组合方式来连接处于串联、并联，乃至网格结构或混合结构的元件；用概率的概念依据元件的可靠性来计算系统的可靠性。

1. 串联结构

从可靠性的观点看，如果系统正常工作取决于组成系统的所有元件都正常工作，则称 n 个元件的集合为串联，元件本身并不必在物理上或拓扑上是串联的。换句话说，只有组成系统的所有元件工作正常，系统才能工作正常，这样的系统被称为非冗余系统，如图4.1所示。

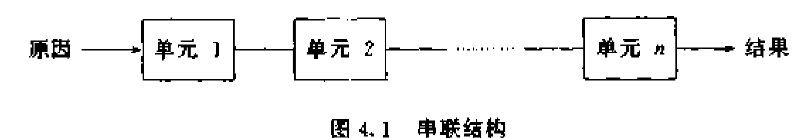


图 4.1　串联结构

令 x_i 表示第 i 个单元工作正常的事件，$\bar{x}_i$ 表示第 i 个单元工作失效的事件，那么系统的可靠性可定义为系统工作正常的概率：

$$\begin{aligned}R &= P(x_1 x_2 \cdots x_n)\\ &= P(x_1)P(x_2|x_1)P(x_3|x_1x_2)\cdots P(x_n|x_1x_2\cdots x_{n-1})\end{aligned}$$

如果单元的失效是相互独立的，则有

$$R = P(x_1)P(x_2)\cdots P(x_n)$$
$$= \prod_{i=1}^{n} P(x_i) \equiv \prod_{i=1}^{n} R_i$$

系统的非可靠性可以表达为

$$Q = P(\bar{x}_1 + \bar{x}_2 + \cdots + \bar{x}_n)$$
$$= 1 - P(x_1 x_2 \cdots x_n)$$
$$= 1 - R$$
$$= 1 - \prod_{i=1}^{n} R_i$$
$$= 1 - \prod_{i=1}^{n} (1 - Q_i)$$

2. 并联结构

从可靠性的观点出发，如果系统中至少有一个元件工作正常，系统就能工作正常，称 n 个元件的集合为并联，这样的系统被称为完全的冗余系统或足够的冗余系统，如图 4.2 所示。

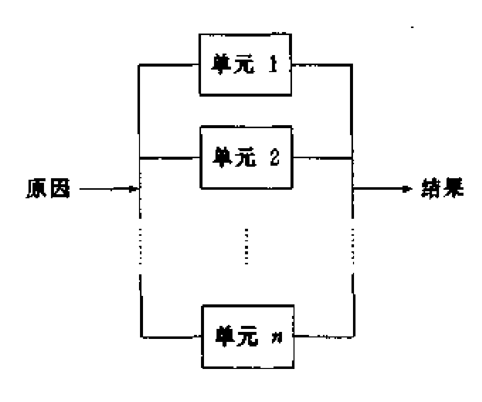


图 4.2 并联结构

系统的可靠性可定义为系统工作正常的概率：

$$R = P(x_1 + x_2 + \cdots + x_n)$$
$$= 1 - P(\bar{x}_1 \bar{x}_2 \cdots \bar{x}_n)$$
$$= 1 - P(\bar{x}_1)P(\bar{x}_2|\bar{x}_1)P(\bar{x}_3|\bar{x}_1\bar{x}_2)\cdots P(\bar{x}_n|\bar{x}_1\bar{x}_2\cdots\bar{x}_{n-1})$$

如果元件的失效是相互独立的，则有

$$R = 1 - \prod_{i=1}^{n} P(\bar{x}_i)$$
$$= 1 - \prod_{i=1}^{n} Q_i$$

系统的非可靠性可以表示为

$$Q = \prod_{i=1}^{n} Q_i$$
$$= \prod_{i=1}^{n} (1 - R_i)$$

3. k/n 结构

在这样的结构中，系统包含 n 个相同的且相互独立的元件，系统要工作正常至少需要其中的 $k<n$ 个元件工作正常。令 p 是每个元件工作正常的概率，恰好 k 个元件工作正常，于是 $(n-k)$ 个元件失效的概率是：

$$\binom{n}{k} p^k(1-p)^{(n-k)}$$

这样的系统称为是有冗余的。对于 $k=1$ 的情形，系统变成真正的并联(完全冗余)系统，而 $k=n$ 时，系统变成串联(非冗余)系统。

4.1.2 多种失效模式的可靠性优化模型

利用冗余元件是提高系统可靠性的易于接受的技术。一般地，这个问题可以描述为非线性整数规划[24,403]。

1. 符号与术语

k/n:F	系统是失效的，当且仅当至少 n 个元件中的 k 个失效
k/n:G	系统是有效的，当且仅当至少 n 个元件中的 k 个有效
O 类失效模式	子系统 i 属 l/m_i: F
A 类失效模式	子系统 i 属 l/m_i: G
m_i	子系统 i 的元件数，$\mathbf{m}=[m_1,m_2,\cdots,m_N]$
s_i	子系统 i 失效模式的总数
h_i	子系统 i 中 O 类失效模式数($u=1,2,\cdots,h_i$)
s_i-h_i	子系统 i 中 A 类失效模式数 ($u=h_i+1, h_i+2, \cdots, s_i$)
q_{iu}	子系统 i 中模式为 u 的每个元件的失效概率
u_i	子系统 i 中元件数 m_i 的上界
b_t	系统资源 t 的可用总量($t=1,2,\cdots,T$)
$g_{ti}(m_i)$	当子系统 i 包含 m_i 个元件时对资源 t 的需求量
$Q_u^O(m_i)$,$Q_u^A(m_i)$	子系统 i(m_i 个元件)中 O 类失效模式或 A 类失效模式的模式 u 的失效概率
$Q^O(m_i)$,$Q^A(m_i)$	子系统 i(m_i 个元件)满足 O 类或 A 类失效模式的失效概率
$Q_i(m_i)$	子系统 i(m_i 元件)的非可靠性
$R(\mathbf{m})$	当元件分配是 $\mathbf{m}$ 时系统的可靠性
$G_t(\mathbf{m})$	当元件分配是 $\mathbf{m}$ 时系统对资源 t 的需求量

2. 数学模型

假定

(1) 子系统 i 的所有元件对于失效模式 u 都是 s(统计)独立的，对于每个 i,u 取自身组合；

(2) 系统关于子系统是 $1/N$:F 型；

(3) 失效模式是失效事件的分割(相互排斥的)，因此，失效模式的失效概率组合就得到总的失效概率；

(4) 在一个子系统内，元件对于某些失效模式是 $1/m_i$:G 型；同时，对于另一些失效模式是 $1/m_i$:F 型。

当整个子系统满足失效条件时，产生失效。在任意时间，子系统 i 的所有元件只能以 m_i 种失效模式的某一种失效，这些失效被分成 O 类和 A 类失效模式。

子系统 i 以 O 类失效模式 u 失效的概率 $Q_u^O(m_i)$ 是

$$Q_u^O(m_i) = 1 - (1 - q_{iu})^{m_i}; \quad u = 1,2, \cdots, h_i \tag{4.1}$$

子系统 i 满足 O 类失效模式的失效概率 $Q^O(m_i)$ 是

$$Q^O(m_i) = \sum_{u=1}^{h_i} Q_u^O(m_i) \tag{4.2}$$

子系统 i 以 A 类失效模式 u 失效的概率 $Q_u^A(m_i)$ 是

$$Q_u^A(m_i) = (q_{iu})^{m_i}; \quad u = h_i + 1, h_i + 2, \cdots, s_i \tag{4.3}$$

子系统 i 满足 A 类失效模式的失效概率 $Q^A(m_i)$ 是

$$Q^A(m_i) = \sum_{u=h_i+1}^{s_i} Q_u^A(m_i) \tag{4.4}$$

子系统 i 的非可靠性 $Q_i(m_i)$ 是 A 类和 O 类的失效概率之和，即

$$Q_i(m_i) = Q^O(m_i) + Q^A(m_i) \tag{4.5}$$

具有多种失效模式的冗余系统可靠性优化问题可以描述为

$$\max \quad R(\mathbf{m}) = \prod_{i=1}^{N} (1 - Q_i(m_i)) \tag{4.6}$$

$$\text{s.t.} \quad G_t(\mathbf{m}) = \sum_{i=1}^{N} g_{ti}(m_i) \leqslant b_t; \quad t = 1,2,\cdots,T \tag{4.7}$$

$$1 \leqslant m_i \leqslant u_i, \text{整数}; \quad i = 1,2,\cdots,N \tag{4.8}$$

优化问题的目的是在满足非线性约束(4.7)和上界(4.8)的非负条件下，确定元件分配，使得非线性系统有最大的可靠性，这是一个 NLP 问题。

4.1.3 可选设计的可靠性优化模型

提高系统可靠性的另一种方式是采用使可靠性增大的比较方式，但费用较高。Fyffe 等人把具有冗余单元和可选设计的可靠性优化问题描述为非线性整数规划[145]。

1. 符号与术语

α_i	第 i 个子系统可选用的设计
β_i	第 i 个子系统可选用设计的上界
$R(\mathbf{m},\boldsymbol{\alpha})$	具有冗余单元 $\mathbf{m}$ 和比较设计 $\boldsymbol{\alpha}$ 的系统可靠性
$C(\mathbf{m},\boldsymbol{\alpha})$	系统满足冗余单元 $\mathbf{m}$ 和可选设计 $\boldsymbol{\alpha}$ 的费用函数
$W(\mathbf{m},\boldsymbol{\alpha})$	系统满足冗余单元 $\mathbf{m}$ 和可选设计 $\boldsymbol{\alpha}$ 的重量函数

C	系统费用的总约束
W	系统重量的总约束

2. 数学模型

假定系统包含 N 个子系统，对于每个子系统，系统设计者有多个可选设计来满足可靠性要求。该问题可以描述为如下非线性整数规划：

$$\max \quad R(\mathbf{m},\boldsymbol{\alpha}) = \prod_{i=1}^{N} R_i(m_i,\alpha_i) \tag{4.9}$$

$$\text{s.t.} \quad C(\mathbf{m},\boldsymbol{\alpha}) = \sum_{i=1}^{N} c_i(\alpha_i)m_i \leqslant C \tag{4.10}$$

$$W(\mathbf{m},\boldsymbol{\alpha}) = \sum_{i=1}^{N} w_i(\alpha_i)m_i \leqslant W \tag{4.11}$$

$$1 \leqslant m_i \leqslant u_i \tag{4.12}$$

$$1 \leqslant \alpha_i \leqslant \beta_i \tag{4.13}$$

$$m_i \text{和 } \alpha_i \text{ 为整数}; \quad i = 1,2,\cdots, N$$

问题的目标是：在系统总费用和总重量不超出允许范围内的条件下，确定选用哪种设计，用多少个冗余单元，使可靠性达到最大。该问题也可以描述为 N 阶段决策问题，其中在每个阶段 i，确定 m_i 和 α_i。

4.2 可靠性优化的简单遗传算法

4.2.1 问题的表述

Gen、Ida 和 Yokata 首次提出了求解可靠性优化的简单遗传算法[159,226,427]。他们考虑了 Tillman 提出的具有多种失效模式的冗余系统的可靠性优化问题[403]，该问题被许多学者用作基准问题[149,158]。问题的目标是在受限于 A 类失效（只有整个系统满足失效条件时）的子系统中具有并联冗余单元的三个非线性约束的条件下使系统可靠性达到最大。问题可表达为如下数学模型：

$$\max \quad R(\mathbf{m}) = \prod_{i=1}^{3}\Big(1 - (1 - (1 - q_{i1})^{m_i+1}) - \sum_{u=2}^{4}(q_{iu})^{m_i+1}\Big)$$

$$\text{s.t.} \quad G_1(\mathbf{m}) = (m_1 + 3)^2 + (m_2)^2 + (m_3)^2 \leqslant 51$$

$$G_2(\mathbf{m}) = 20\sum_{i=1}^{3}(m_i + \exp(-m_i)) \geqslant 120$$

$$G_3(\mathbf{m}) = 20\sum_{i=1}^{3}(m_i \exp(-m_i/4)) \geqslant 65$$

$$1 \leqslant m_1 \leqslant 4, \quad 1 \leqslant m_2, \quad m_3 \leqslant 7$$

$$m_i \geqslant 0 \text{ ，整数}; \quad i = 1,\cdots, 3$$

其中，$\mathbf{m}=[m_1, m_2, m_3]$。对于子系统受限于一种 O 类失效（$h_i=1$）和三种 A 类失效的四种失效模式（$s_i= 4$），$i=1,2,3$，每个子系统的失效概率如表 4.1 所示。

表 4.1 每个子系统的失效模式和概率

子系统 i	失效模式 $s_i=4, h_i=1$	失效概率 q_{iu}
1	O	0.01
	A	0.05
	A	0.10
	A	0.18
2	O	0.08
	A	0.02
	A	0.15
	A	0.12
3	O	0.04
	A	0.05
	A	0.20
	A	0.10

4.2.2 遗传算法与计算实例

1. 染色体表达

传统的遗传算法中，染色体用二进制串表达。对于该问题，变量 m_i 的整数值用二进制串表达，串的长度依赖于冗余单元的上界 u_i。例如，当 $u_i=4$ 时，需要长度为 3 的二进位表示 m_i。在本例中，每个子系统冗余单元的上界 $u_1=4, u_2=7, u_3=7$，因此决策变量 m_i 需要 3 位二进位，就是说总共需要 9 位。如果 $m_1=2, m_2=3, m_3=3$，我们就有如下的染色体：

$$
\begin{aligned}
\mathbf{v} &= [x_{33}\ x_{32}\ x_{31}\ x_{23}\ x_{22}\ x_{21}\ x_{13}\ x_{12}\ x_{11}] \\
&= [0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0]
\end{aligned}
$$

其中，x_{ij}是变量 m_i 的第 j 位二进位。

2. 初始种群

染色体的初始种群随机产生，每个染色体是一个 9 位的二进制串。这样，产生的染色体可能由于违背系统约束或/和超过上界而不可行。因此需要通过可行性检查这一步以保证所有的染色体是可行的。

```
初始化过程
begin
for i ← 1 to pop_size do
  随机产生染色体 v_i;
  if v_i 非可行 then
    i ← i − 1;
  end
```

```
    end
end
```

令 $pop_size=5$，随机产生如下染色体：

$$v_1=[1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1]$$
$$v_2=[1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1]$$
$$v_3=[1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0]$$
$$v_4=[0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1]$$
$$v_5=[1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1]$$

相应的整数值是

	m_3	m_2	m_1
$v_1=$	[6	2	1]
$v_2=$	[4	1	3]
$v_3=$	[5	1	2]
$v_4=$	[3	2	3]
$v_5=$	[4	2	1]

3. 染色体的评估

评价染色体时，对于每个可行的染色体，赋给目标函数值 $R(\mathbf{m})$，而对于每个非可行的染色体，给予一个很大的惩罚：

$$eval(v_k)=\begin{cases}R(\mathbf{m}); & G_t(\mathbf{m})\leqslant b_t,\ \forall t;\ 1\leqslant m_i\leqslant u_i,\ \forall i\\ -M; & \text{其他}\end{cases}$$

其中，v_k 代表第 k 个染色体；M 是一个很大的正整数。

染色体的适值，即系统可靠性为

$$eval(v_1)=0.543625$$
$$eval(v_2)=0.632703$$
$$eval(v_3)=0.610062$$
$$eval(v_4)=0.629119$$
$$eval(v_5)=0.589642$$

4. 交叉

利用单点交叉法。令交叉概率 $p_c=0.4$，随机数序列为

0.550279，0.379650，0.243294，0.494583，0.771811

如果随机数 $r<p_c$，选择相关的染色体进行交叉，即选择染色体 v_2 和 v_3 进行交叉。切点的位置在[1,9]中随机产生，假定位置是 3，有

$$v_2=[1\ 0\ 0\mid 0\ 0\ 1\ 0\ 1\ 1]$$
$$\updownarrow$$
$$v_3=[1\ 0\ 1\mid 0\ 0\ 1\ 0\ 1\ 0]$$

交换双亲的右部后产生的后代是

$$\mathbf{o}_1 = [\,1\,0\,0\,0\,0\,1\,0\,1\,0\,]$$
$$\mathbf{o}_2 = [\,1\,0\,1\,0\,0\,1\,0\,1\,1\,]$$

后代的相应整数值是

$$\begin{array}{lccc} & m_3 & m_2 & m_1 \\ \mathbf{o}_1 = [& 4 & 1 & 2\,] \\ \mathbf{o}_2 = [& 5 & 1 & 3\,] \end{array}$$

适值为

$$eval(\mathbf{o}_1) = 0.607591$$
$$eval(\mathbf{o}_2) = 0.635277$$

5. 变异

变异以位为基础进行。令变异的概率为 $p_m = 0.1$，也就是说，平均有10%的位数进行变异。每代中，总的种群有 $9 \times 5 = 45$ 位，其中4或5位要作变异。随机数 r 在[0,1]中产生；如果 $r < 0.1$，则变异相关的位。

总共产生45个随机数，其中4个小于0.1。位数和对应的染色体及染色体中的位如下：

位数	染色体	位置	随机数
14	2	5	0.018754
15	2	6	0.094512
18	2	9	0.065742
31	4	4	0.001257

从表中选取染色体 $\mathbf{v}_2$ 和 $\mathbf{v}_4$ 进行变异，变异的结果为

$$\mathbf{v}_2 = [\,1\,0\,0\,0\,0\,1\,0\,1\,1\,]$$
$$\updownarrow\updownarrow\quad\updownarrow$$
$$\mathbf{o}_3 = [\,1\,0\,0\,0\,1\,0\,0\,1\,0\,]$$
$$\mathbf{v}_4 = [\,0\,1\,1\,0\,1\,0\,0\,1\,1\,]$$
$$\updownarrow$$
$$\mathbf{o}_4 = [\,0\,1\,1\,1\,1\,0\,0\,1\,1\,]$$

相应的整数值为

$$\begin{array}{lccc} & m_3 & m_2 & m_1 \\ \mathbf{o}_3 = [& 4 & 2 & 2\,] \\ \mathbf{o}_4 = [& 3 & 5 & 3\,] \end{array}$$

适值为

$$eval(\mathbf{o}_3) = -M$$
$$eval(\mathbf{o}_4) = -M$$

这表示两个染色体都是非可行染色体。

6. 选择

采用确定性的选择策略，也就是说，按照降序排列所有双亲和后代中的个体，选择前 *pop_size* 个染色体组成新一代种群。

$$\mathbf{v}'_1 = [\,1\,0\,1\,0\,0\,1\,0\,1\,1\,] \quad (\mathbf{o}_2), \quad eval(\mathbf{v}'_1) = 0.635277$$
$$\mathbf{v}'_2 = [\,1\,0\,0\,0\,0\,1\,0\,1\,1\,] \quad (\mathbf{v}_2), \quad eval(\mathbf{v}'_2) = 0.632703$$
$$\mathbf{v}'_3 = [\,0\,1\,1\,0\,1\,0\,0\,1\,1\,] \quad (\mathbf{v}_4), \quad eval(\mathbf{v}'_4) = 0.629119$$
$$\mathbf{v}'_4 = [\,1\,0\,1\,0\,0\,1\,0\,1\,0\,] \quad (\mathbf{v}_3), \quad eval(\mathbf{v}'_3) = 0.610062$$
$$\mathbf{v}'_5 = [\,1\,0\,0\,0\,0\,1\,0\,1\,0\,] \quad (\mathbf{o}_1), \quad eval(\mathbf{v}'_1) = 0.607591$$

当代数为 50 时，遗传算法随机运行的最好结果是

$$\mathbf{v}^* = [\,0\,1\,1\,0\,0\,1\,0\,1\,0\,], \quad eval(\mathbf{v}^*) = 0.660685$$

相应的整数值为

$$[m_3\,m_2\,m_1] = [\,3\,1\,2\,]$$

本结果与 Tillman[403]和 Gen[149]给出的结果一致。30 次运行的统计数据如表 4.2所示。

表 4.2　30 次实验的统计数据

总次数	30
达到最优的频率	0.933
达到最优的最早代数	1
达到最优的最迟代数	44
达到最优的平均代数	13.4
标准方差	0.00093

注：标准方差：$(x^* - \sum x_i/N)/x^*$ 。

4.3　冗余单元和可选设计的可靠性优化

4.3.1　问题的表述

Gen，Yokota，Ida ，Taguchi 把一般可靠性优化问题进一步扩展到具有冗余单元和可选设计的可靠性优化问题[159,428]。因为此问题比前面讨论的问题更为复杂，典型的遗传算法必须经过修改才能有效。

这里用到的实例见 Fyffe 等人的论文[145]：

$$\max \quad R(\mathbf{m},\boldsymbol{\alpha}) = \prod_{i=1}^{14} (\,1 - (1 - R_i(\alpha_i))^{m_i}) \tag{4.14}$$

$$\text{s.t.} \quad G_1(\mathbf{m},\boldsymbol{\alpha}) = \sum_{i=1}^{14} c_i(\alpha_i) m_i \leqslant 130 \tag{4.15}$$

$$G_2(\mathbf{m},\boldsymbol{\alpha}) = \sum_{i=1}^{14} w_i(\alpha_i) m_i \leqslant 170 \tag{4.16}$$

$$1 \leqslant m_i \leqslant u_i, \quad \forall i \tag{4.17}$$

$$1 \leqslant \alpha_i \leqslant \beta_i, \quad \forall i \tag{4.18}$$

$$m_i,\ \alpha_i \geqslant 0,\ \text{整数},\ \forall i$$

其中，α_i 表示第 i 个子系统的可选设计；m_i 表示用于第 i 个子系统相同的冗余单元；u_i 是第 i 个子系统冗余单元的上界；β_i 表示第 i 个子系统可选设计的上界。问题是在系统总费用和总重量的允许范围内，确定选用哪个设计，用多少个冗余单元即可达到最大的系统可靠性。数据如表 4.3 所示。

表 4.3 冗余单元与可选设计

子系统 i	可选设计											
	1			2			3			4		
	R	c_i	w_i	R	c_i	w_i	R	c_i	w_i	R	c_i	w_i
1	0.90	1	3	0.93	1	4	0.91	2	2	0.95	2	5
2	0.95	2	8	0.94	1	10	0.93	1	9	*	*	*
3	0.85	2	7	0.90	3	5	0.87	1	6	0.92	4	4
4	0.83	3	5	0.87	4	6	0.85	5	4	*	*	*
5	0.94	2	4	0.93	2	3	0.95	3	5	*	*	*
6	0.99	3	5	0.98	3	4	0.97	2	5	0.96	2	4
7	0.91	4	7	0.92	4	8	0.94	5	9	*	*	*
8	0.81	3	4	0.90	5	7	0.91	6	6	*	*	*
9	0.97	2	8	0.99	3	9	0.96	4	7	0.91	3	8
10	0.83	4	6	0.85	4	5	0.90	5	6	*	*	*
11	0.94	3	5	0.95	4	6	0.96	5	6	*	*	*
12	0.79	2	4	0.82	3	5	0.85	4	6	0.90	5	7
13	0.98	2	5	0.99	3	5	0.97	2	6	*	*	*
14	0.90	4	6	0.92	4	7	0.95	5	6	0.99	6	9

注：对 $i = 1,3,6,9,12,14, \beta_i=4$；对 $i = 2,4,5,7,8,10,11,13, \beta_i=3$。$u_i=5, \forall i$。

4.3.2 遗传算法与计算实例

1. 染色体表达

用可选设计 α_{ki} 和冗余单元 m_{ki} 的有序对定义基因：

$$v_{ki} = (\alpha_{ki},\ m_{ki})$$

其中，下标 k 表示基因所属的染色体的标号；i 表示子系统 i。染色体就是这些基因的有序表：

$$\mathbf{v}_k = [\, v_{k1} \quad v_{k2} \quad \cdots \quad v_{k14} \,]$$

也可表示为

$$\mathbf{v}_k = [\, (\alpha_{k1} \quad m_{k1}) \quad (\alpha_{k2} \quad m_{k2}) \quad \cdots \quad (\alpha_{k14} \quad m_{k14}) \,]$$

2. 初始种群

按如下方式随机产生染色体的初始种群：

```
初始化过程
begin
for k ← 1 to pop_size do
  for i ← 1 to 14 do
    α_ki ← random(1,β_i);
    m_ki ← random(1,u_i);
  end
  v_k ← [ (α_k1, m_k1) … (α_k14, m_k14) ];
end
end
```

其中，$random(1,num)$返回一个$[1,num]$区间的随机整数。令 $pop_size=8$，随机产生的染色体为

$\mathbf{v}_1=[\,(2,1)(1,1)(4,1)(1,4)(3,4)(4,2)(2,1)(1,3)(3,2)(2,2)(2,1)(1,2)(3,4)(3,4)\,]$

$\mathbf{v}_2=[\,(2,2)(1,3)(2,3)(1,1)(2,5)(2,1)(1,1)(3,1)(4,1)(3,2)(1,2)(1,2)(1,2)(1,1)\,]$

$\mathbf{v}_3=[\,(1,3)(2,3)(3,1)(3,2)(2,3)(1,1)(2,1)(3,1)(4,1)(3,4)(1,2)(1,2)(3,4)(2,1)\,]$

$\mathbf{v}_4=[\,(2,4)(3,1)(4,3)(3,1)(1,1)(2,3)(3,2)(2,1)(3,3)(2,2)(3,5)(1,1)(3,1)(1,2)\,]$

$\mathbf{v}_5=[\,(2,2)(2,1)(2,4)(1,3)(1,2)(2,5)(1,2)(1,3)(2,1)(2,2)(2,2)(3,1)(2,1)(2,3)\,]$

$\mathbf{v}_6=[\,(1,1)(3,4)(2,3)(3,1)(1,4)(1,1)(3,1)(1,4)(1,1)(3,1)(1,1)(2,5)(3,1)(2,1)\,]$

$\mathbf{v}_7=[\,(4,1)(3,1)(2,4)(3,2)(3,4)(4,3)(2,1)(3,4)(4,2)(2,4)(1,1)(1,2)(1,1)(2,1)\,]$

$\mathbf{v}_8=[\,(3,2)(3,1)(3,3)(2,2)(2,3)(4,2)(3,2)(3,3)(3,3)(3,1)(2,1)(1,3)(3,2)(1,2)\,]$

3. 染色体评估

随机方法产生初始种群，可能导致一些违反系统约束的不可行染色体。事实上，在应用遗传算法求解大规模整数优化问题时，遗传算子产生的染色体或随机产生的初始染色体常常违反给定问题的约束，这是常见而又关键的问题。处理这种非可行性的现成方法，是前面讨论过的给那些非可行的染色体以大的惩罚。这种方法本质上是在进化过程中淘汰非可行解而缩小搜索空间。这种选择机理很难找到全局最好的候选解，遗传搜索将失去效率。

为了克服这个问题，引进一个特殊的度量函数来评价非可行的染色体脱离可行域的程度。一般地，最优解常常在可行域和非可行域的边界达到。当我们只是简单地赋给每个非可行的染色体一个大的常数惩罚时，它们将从进化过程中被剔除，遗传搜索将仅从可行域一侧趋于最优解。如果将非可行解的信息嵌入到适应性评价中，遗传搜索将从可行域和非可行域两侧趋于最优解，这样搜索可在更大的空间中进行，如图 4.3 所示。

按下式可度量非可行染色体的非可行性的程度：

$$d_{ki}=\begin{cases}0, & G_i(\mathbf{m},\boldsymbol{\alpha})\leqslant b_i\\ (G_i(\mathbf{m},\boldsymbol{\alpha})-b_i)/b_i, & \text{其他}\end{cases}$$

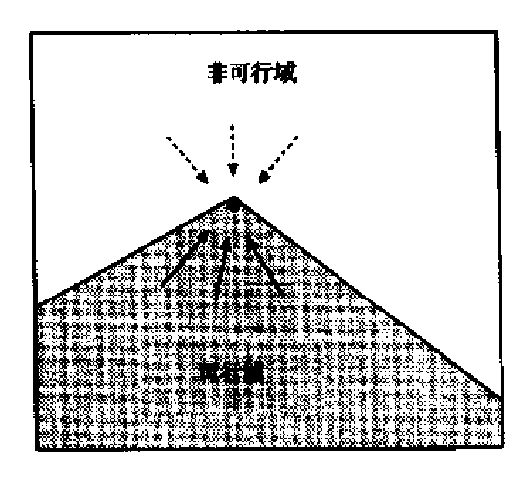


图 4.3 遗传搜索方向

其中，下标 k 表示第 k 个染色体，t 表示第 t 个约束，适值函数为

$$eval(\mathbf{v}_k) = R(\mathbf{m},\boldsymbol{\alpha})\left(1 - \frac{1}{T}\sum_{t=1}^{T} d_{kt}\right)$$

其中，T 是约束的总数。

初始染色体的适值为

$$eval(\mathbf{v}_1) = 0.656578,\quad eval(\mathbf{v}_5) = 0.741668$$
$$eval(\mathbf{v}_2) = 0.516839,\quad eval(\mathbf{v}_6) = 0.520029$$
$$eval(\mathbf{v}_3) = 0.561119,\quad eval(\mathbf{v}_7) = 0.637986$$
$$eval(\mathbf{v}_4) = 0.493847,\quad eval(\mathbf{v}_8) = 0.753369$$

4. 交叉

对于组合优化问题，Syswerda 提出的均匀交叉算子要优于传统的交叉策略[391]。均匀交叉首先产生随机的交叉罩，然后根据交叉罩交换相关的基因。交叉罩只是有相同染色体数量的二进位串。交叉罩中每位的奇偶性确定了对于后代中每个相应的位从哪个父代得到，如图 4.4 所示。

令交叉率 $p_c = 0.4$，随机数序列是

0.486129　0.536149　0.785178　0.754631

0.105411　0.084658　0.105411　0.771811

如果随机数 $r < p_c$，选择相关的染色体进行交叉，本例中选取染色体 $\mathbf{v}_6$ 和 $\mathbf{v}_7$ 交叉。随机产生的位置是 12，8 和 2，于是有

$$\mathbf{v}_6 = [\,(1,1)(3,4)(2,3)(3,1)(1,4)(1,1)(3,1)(1,4)(1,1)(3,1)(1,1)(2,5)(3,1)(2,1)\,]$$

$$\updownarrow \qquad\qquad \updownarrow \qquad\qquad \updownarrow$$

$$\mathbf{v}_7 = [\,(4,1)(3,1)(2,4)(3,2)(3,4)(4,3)(2,1)(3,4)(4,2)(2,4)(1,1)(1,2)(1,1)(2,1)\,]$$

交换相关的基因产生的后代是

$$\mathbf{o}_1 = [\,(1,1)(3,1)(2,3)(3,1)(1,4)(1,1)(3,1)(3,4)(1,1)(3,1)(1,1)(1,2)(3,1)(2,1)\,]$$

$$\mathbf{o}_2 = [\,(4,1)(3,4)(2,4)(3,2)(3,4)(4,3)(2,1)(1,4)(4,2)(2,4)(1,1)(2,5)(1,1)(2,1)\,]$$

适值为

$$eval(\mathbf{o}_1) = 0.462970$$
$$eval(\mathbf{o}_2) = 0.653214$$

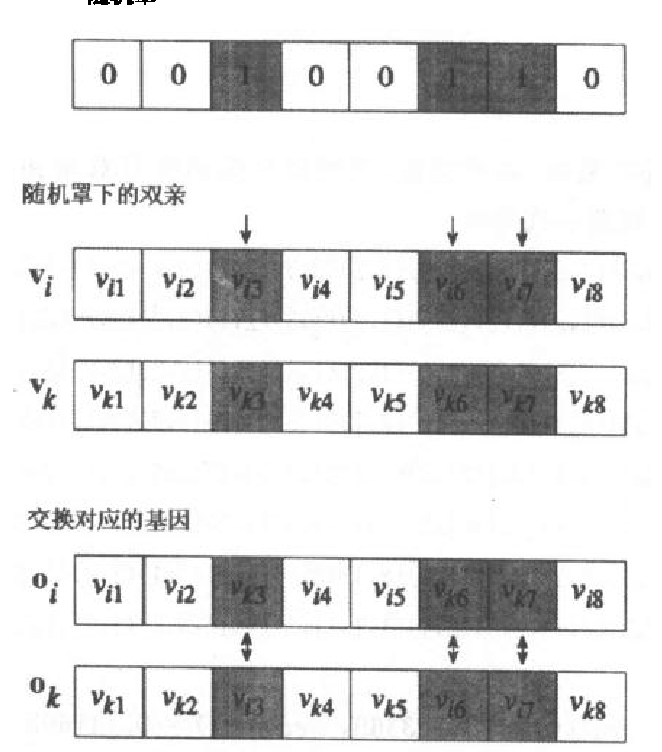


图 4.4　均匀交叉算子图解

5. 变异

采用随机摄动进行变异。对于一个选取变异的基因 $v_{ki}=(\alpha_{ki}, m_{ki})$，$\alpha_{ki}$用$[1,\beta_i]$中的随机整数来代替，$m_{ki}$用$[1,u_i]$中的随机整数代替。

令变异率为 $p_m=0.05$，也就是说，平均有 $14\times8\times0.05=5.6$ 位要变异。在产生的112个随机数中有4个小于0.05。位数和相应的染色体及染色体中位的位置如下表所示：

位数	染色体	位置	随机数
22	2	8	0.031892
31	3	3	0.020447
43	4	1	0.041556
70	5	14	0.001257

从表中选取染色体 $\mathbf{v}_2$，$\mathbf{v}_3$，$\mathbf{v}_4$ 和 $\mathbf{v}_5$ 进行变异，产生的后代为

$\mathbf{o}_3=[(2,2)(1,3)(2,3)(1,1)(2,5)(2,1)(1,1)(1,4)(4,1)(3,2)(1,2)(1,2)(1,2)(1,1)]$

$\mathbf{o}_4=[(1,3)(2,3)(1,5)(3,2)(2,3)(1,1)(2,1)(3,1)(4,1)(3,4)(1,2)(1,2)(3,4)(2,1)]$

$\mathbf{o}_5=[(3,4)(3,1)(4,3)(3,1)(1,1)(2,3)(3,2)(2,1)(3,3)(2,2)(3,5)(1,1)(3,1)(1,2)]$

$\mathbf{o}_6=[(2,2)(2,1)(2,4)(1,3)(1,2)(2,5)(1,2)(1,3)(2,1)(2,2)(2,2)(3,1)(2,1)(4,1)]$

其适值为

$$eval(\mathbf{o}_3)=0.567215$$

$$eval(\mathbf{o}_4)=0.613955$$

$$eval(\mathbf{o}_5) = 0.493827$$
$$eval(\mathbf{o}_6) = 0.734647$$

6. 选择

采用确定性的选择策略，也就是说，按照降序排列所有双亲和后代的个体，选择前 *pop_size* 个染色体组成新一代种群。

$\mathbf{v}_1' = [(3,2)(3,1)(3,3)(2,2)(2,3)(4,2)(3,2)(3,3)(3,3)(3,1)(2,1)(1,3)(3,2)(1,2)]$ $(\mathbf{v}_6)$

$\mathbf{v}_2' = [(2,2)(2,1)(2,4)(1,3)(1,2)(2,5)(1,2)(1,3)(2,1)(2,2)(2,2)(3,1)(2,1)(2,3)]$ $(\mathbf{v}_5)$

$\mathbf{v}_3' = [(2,2)(2,1)(2,4)(1,3)(1,2)(2,5)(1,2)(1,3)(2,1)(2,2)(2,2)(3,1)(2,1)(4,1)]$ $(\mathbf{o}_6)$

$\mathbf{v}_4' = [(2,1)(1,1)(4,1)(1,4)(3,4)(4,2)(2,1)(1,3)(3,2)(2,2)(2,1)(1,2)(3,4)(3,4)]$ $(\mathbf{v}_1)$

$\mathbf{v}_5' = [(4,1)(3,4)(2,4)(3,2)(3,4)(4,3)(2,1)(1,4)(4,2)(2,4)(1,1)(2,5)(1,1)(2,1)]$ $(\mathbf{o}_2)$

$\mathbf{v}_6' = [(4,1)(3,1)(2,4)(3,2)(3,4)(4,3)(2,1)(3,4)(4,2)(2,4)(1,1)(1,2)(1,1)(2,1)]$ $(\mathbf{v}_7)$

$\mathbf{v}_7' = [(1,3)(2,3)(1,5)(3,2)(2,3)(1,1)(2,1)(3,1)(4,1)(3,4)(1,2)(1,2)(3,4)(2,1)]$ $(\mathbf{o}_4)$

$\mathbf{v}_8' = [(2,2)(1,3)(2,3)(1,1)(2,5)(2,1)(1,1)(1,4)(4,1)(3,2)(1,2)(1,2)(1,2)(1,1)]$ $(\mathbf{o}_3)$

新一代的适值为

$$eval(\mathbf{v}_1') = 0.753369,\quad eval(\mathbf{v}_2') = 0.741668$$
$$eval(\mathbf{v}_3') = 0.734647,\quad eval(\mathbf{v}_4') = 0.656578$$
$$eval(\mathbf{v}_5') = 0.653214,\quad eval(\mathbf{v}_6') = 0.637986$$
$$eval(\mathbf{v}_7') = 0.613955,\quad eval(\mathbf{v}_8') = 0.567215$$

随机运行中在第 623 代产生的最好染色体为

$\mathbf{v}^* = [(3,3)\ (1,2)\ (4,3)\ (3,3)\ (2,3)\ (2,2)\ (1,2)\ (1,4)\ (3,2)\ (2,3)\ (1,2)\ (1,4)\ (2,2)\ (3,2)]$

系统的可靠性是 0.970015，这与 Nakagawa 等人[310]和 Gen 等人[156]的结果是一致的。30 次运行的统计数据如表 4.4 所示。

表 4.4 30 次实验的统计数据

总运行数	30
达到最优解的频率	0.170
达到最优解的最早代数	161
达到最优解的最迟代数	975
达到最优解的平均代数	423.8
标准方差	0.002375

注：标准方差：$\left(x^* - \sum x_i/N\right)/x^*$。

4.4 冗余混合元件的可靠性优化

4.4.1 问题的描述

Coit 和 Smith 研究了子系统中有冗余混合元件的可靠性优化问题，他们把总系统分解成并联的 N 个子系统[82,83]。对于每个需求函数，有多种不同类型的元件，它们有

不同的费用、可靠性、重量及其他性质。每个子系统被看作是混合元件组成的，并有 k/n:G 冗余。这种冗余分配问题就是在满足总的重量和费用约束的条件下找出最优的元件组合和冗余水平，使得系统具有最大的可靠性，问题可以表述为如下模型：

$$\max \quad \prod_{i=1}^{N} R_i(\mathbf{x}_i \mid k_i) \tag{4.19}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} C_i(\mathbf{x}_i) \leqslant C \tag{4.20}$$

$$\sum_{i=1}^{N} W_i(\mathbf{x}_i) \leqslant W \tag{4.21}$$

$$k_i \leqslant \sum_{j=1}^{m_i} x_{ij} \leqslant u_i; \quad i = 1,2,\cdots,N \tag{4.22}$$

$$x_{ij} \text{ 为整数}; \quad \forall\, i,j$$

其中：

N——子系统的数目；

C——费用约束；

W——重量约束；

x_{ij}——子系统 i 中 j 种元件的数目 $\mathbf{x}_i=(x_{i1},x_{i2},\cdots,x_{im_i})$；

m_i——子系统 i 的可选用的元件数；

k_i——子系统 i 的并联元件数下界；

u_i——子系统 i 的并联元件数上界；

$R_i(\mathbf{x}_i \mid k_i)$——对于给定的 k_i，子系统 i 的可靠性；

$C_i(\mathbf{x}_i)$——子系统 i 的总费用；

$W_i(\mathbf{x}_i)$——子系统 i 的总重量。

显然，子系统 i 的总费用和总重量可以表示为

$$C_i(\mathbf{x}_i)= \sum_{j=1}^{m_i} c_{ij}\, x_{ij} \tag{4.23}$$

$$W_i(\mathbf{x}_i)= \sum_{j=1}^{m_i} w_{ij}\, x_{ij} \tag{4.24}$$

其中：

c_{ij}——子系统 i 的可用元件 j 的费用；

w_{ij}——子系统 i 的可用元件 j 的重量。

系统可靠性可以表示为决策变量 x_{ij} 的函数：

$$\prod_{i=1}^{N} R_i(\mathbf{x}_i \mid k_i) = \prod_{i=1}^{N}\left(1 - \sum_{l=0}^{k_i-1} \sum_{\mathbf{t}\in T_l} \prod_{j=1}^{m_i} \binom{x_{ij}}{t_j} r_{ij}^{t_j}(1 - r_{ij})^{x_{ij}-t_j}\right) \tag{4.25}$$

其中：

T_l——集合 $\left\{(t_1,t_2,\cdots,t_{m_i}) \mid \sum_{j=1}^{m_i} t_j= l\right\}$；

$\mathbf{t}$——向量 $(t_1,t_2,\cdots,t_{m_i})$；

r_{ij}——子系统 i 中可用元件 j 的可靠性。

4.4.2 遗传算法与计算实例

1. 解的编码

冗余分配问题的每个可能解是 N 个不同子系统 $n_i(k_i \leqslant n_i \leqslant u_i)$ 的并联元件的集合。n_i 个元件可以从 m_i 个可用元件的任意组合中选择。m_i 个元件按其可靠性下降的顺序标注，即 1 表示最可靠的元件等。解的编码是用 $\sum u_i$ 个位置的整数向量表示。每个子系统根据其组成元件的可靠性用 u_i 个位置表示，下标 m_i+1 分配给一个没有使用附加元件的空位置。各子系统相邻排列以便用向量来表达。例如，考虑 $N=3$，$m_1=5$，$m_2=4$，$m_3=5$ 的系统，对于所有的 i，预先假定 $u_i=5$，即

$$v_k = [1\ 1\ 6\ 6\ 6 \mid 2\ 2\ 3\ 5\ 5 \mid 4\ 6\ 6\ 6\ 6]$$

表示一个预期的解。在这个解中第一个子系统有两个最可靠的并联元件；两个次最可靠的元件和一个次次可靠的元件并联在第二个子系统中；一个第四可靠的元件在第三个子系统中。

2. 初始种群

随机选取初始种群，对于每个解向量，k_i 与 u_i 间的整数 N 代表每个特定子系统处于并联的部件数(n_i)。假定每种类型元件的可用需求量是无限的，那么 n_i 个部件从 m_i 个元件中随机均匀地选取，选取的元件依其可靠性排序，根据经验选取初始种群的规模为 40。

3. 交叉

本节选用均匀交叉，双亲的公共基因位的值在后代中被保留下来，而非公共基因位的值则以相等的概率从双亲的个体中选取。

4. 变异

采用随机摄动进行变异。每个随机选取作为变异的解向量以等同于变异率的概率改变。变异的元件以 50%的概率变为下标 m_i+1，以 50%的概率变为从 m_i 个候选元件中随机选取的元件。

5. 选择

这里指采用保存精英的选择策略。经过交叉繁殖，*pop_size* 个双亲和后代的最好解被保存下来组成新的种群，然后从种群中选一个非优解进行变异。后代中的最好解永远不会被选取进行变异，以确保最好解不会通过变异而改变。这是保存精英的基本形式[186]。

6. 自适应惩罚

这里用到的惩罚函数是基于对应于约束集的近似可行阈 NFT 概念构造的。惩罚函数表述为给定解与可行域的“距离”的非增函数。惩罚函数能使算法在可行域和可行域的 NFT 邻域内搜索，并越过近似可行阈的搜索予以惩罚。冗余分配问题的惩罚目标函数

如下：

$$R_{ip} = R_i - \left(\left(\frac{\Delta w_i}{\mathrm{NFT}_w} \right)^{\alpha} + \left(\frac{\Delta c_i}{\mathrm{NFT}_c} \right)^{\alpha} \right) (R_{\mathrm{all}} - R_{\mathrm{feas}}) \tag{4.26}$$

其中：

R_{ip}——解 i 的惩罚目标函数值；

R_i——解 i 的非惩罚目标函数值；

R_{all}——最好解的非惩罚值；

R_{feas}——最好可行解的值；

NFT_w——重量约束的近似可行临界值；

NFT_c——费用约束的近似可行临界值；

Δw_i——解 i 的超重量值；

Δc_i——解 i 的超费用值；

α——用户指定的重要性参数。

对于一个给定的问题和约束集，有时不能指出收敛到最优或邻近最优的可行解所需要的搜索的程度。尽管可以通过实验找到对于任意特定冗余分配问题的 NFT 的有效值，但用于确定特定问题实例的惩罚函数仍是不理想的。为此，近年来提出了包括动态 NFT 在内的各种近似确定 NFT 的规则。

动态 NFT 定义为

$$\mathrm{NFT} = \frac{\mathrm{NFT}_0}{1 + \lambda g} \tag{4.27}$$

其中，NFT_0 是 NFT 的上界或初始点；g 是迭代数；λ 是确保 NFT_0 和 0 之间的整个域均被搜索到的任意常数。

值得注意的是自适应项($R_{\mathrm{all}} - R_{\mathrm{feas}}$)可能造成这种惩罚方法的两种意外情形：0 惩罚和超惩罚。对于 $R_{\mathrm{all}} = R_{\mathrm{feas}}$ 的情形，即使存在非可行解，这种方法将对所有的非可行解给予 0 惩罚。在进化过程初期出现大值的初始解 R_{all} 时，本方法将对所有的非可行解给予超惩罚。

7. 测试问题及结果

测试问题选自 Fyffe，Hines 和 Lee 的文章[145]，其中 33 个不同的问题源于 Nakagawa 和 Miyazaki 的文章[310]。对于有 14 个子系统，每个子系统有 3 或 4 个可供选择的元件，且 $k=1$ 对所有子系统，问题的目标是系统的可靠性达到最大。对于每个元件，则有相应的可靠性、费用和重量。在原问题中，只允许相同的元件处于并联的位置。在 Smith 和 Coit 的研究中，则是通过在子系统中分配冗余混合元件来达到系统总的最大的可靠性。

用 GA 在不同 NFT 值下，分析了 33 个问题的变化，包括

(1) 静态 NFT(每个约束的 5%，3%和 1%)；

(2) 动态 NFT；

(3) 零 NFT(仅允许可行)。

对于 33 个不同问题和不同 NFT 的选择量做了 10 次运行，其结果如表 4.5 和表 4.6 所示。表 4.5 给出了对于 5 个 NFT 中的每个选择量，GA 收敛到最终可行解的百分率。另

外，表中还给出了 GA 搜索达到最好可行解的问题的百分率。该结果好于 Nakagawa 和 Miyazaki (N&M)的结果。表 4.6 给出了 33 个问题的最好可行解、平均可行解和平均标准偏差。

表 4.5 可行性和性能的比较

百分比	0NFT	5%NFT	3%NFT	1%NFT	动态 NFT
可行性	100.00%	1.21%	80.00%	100.00%	100.00%
最好>N&M	0.00%	63.64%	45.45%	9.09%	81.82%
全部>N&M	0.00%	27.27%	15.45%	0.91%	44.85%

表 4.6 最终解的比较

性能	0NFT	5%NFT	3%NFT	1%NFT	动态 NFT
最好	0.97096	0.97337	0.97302	0.97180	0.97366
平均	0.96894	0.97239	0.97167	0.96956	0.97288
偏差 (%)	0.14933	0.08218	0.11305	0.15847	0.06573

4.5 模糊目标和模糊约束的可靠性优化

4.5.1 问题的描述

Sasaki 和 Gen 研究了具有模糊目标和模糊约束的可靠性优化问题[367]。其相应的清晰问题是有多种失效模式的冗余系统的最优设计问题，见 4.1.2 节。

模糊目标和模糊约束的冗余系统的可靠性优化可表述为

$$\max \quad R(\mathbf{m}) = \prod_{i=1}^{N}(1 - Q_i(m_i)) \gtrsim g_0 \tag{4.28}$$

$$\text{s.t.} \quad G_t(\mathbf{m}) = \sum_{i=1}^{N} g_{ti}(m_i) \lesssim b_t; \quad t = 1,2,\cdots,T \tag{4.29}$$

$$1 \leqslant m_i \leqslant u_i; \quad i = 1,2,\cdots,N \tag{4.30}$$

符号$\lesssim$和$\gtrsim$代表模糊不等式，g_0 是由决策者给定的可靠性 $R(\mathbf{m})$的期望目标值，其余的符号意义见前节。

令 z_0^- 是系统可靠性的最坏容许值，目标(4.28)的隶属函数 $\mu_0(\mathbf{m})$定义为

$$\mu_0(\mathbf{m}) = \begin{cases} 1, & R(\mathbf{m}) > g_0 \\ \dfrac{R(\mathbf{m}) - z_0^-}{g_0 - z_0^-}, & z_0^- \leqslant R(\mathbf{m}) \leqslant g_0 \\ 0, & R(\mathbf{m}) < z_0^- \end{cases} \tag{4.31}$$

如图 4.5 所示。

令 δ_t 是 t 种资源的允许超额使用量，约束(4.29)的隶属函数 μ_t 定义为

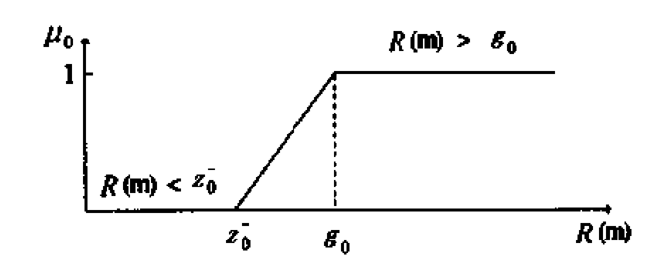


图 4.5 隶属函数 $\mu_0(\mathbf{m})$的图解

$$\mu_t(\mathbf{m}) = \begin{cases} 1, & G_t(\mathbf{m}) < b_t \\ 1 - \dfrac{G_t(\mathbf{m}) - b_t}{\delta_t}, & b_t \leqslant G_t(\mathbf{m}) \leqslant b_t + \delta_t \\ 0, & G_t(\mathbf{m}) > b_t + \delta_t \end{cases} \tag{4.32}$$

如图 4.6 所示。

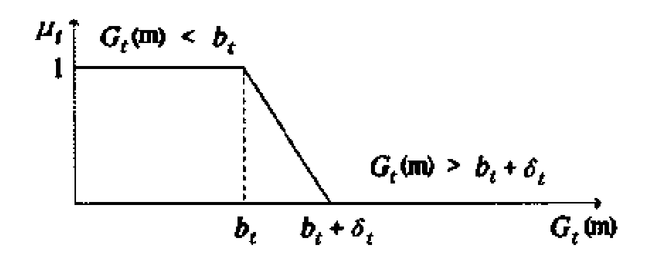


图 4.6 隶属函数 $\mu_t(\mathbf{m})$的图解

于是模糊可靠性优化问题(4.28)～(4.30)可转化为如下等价的清晰问题：

$$\max \quad f(\mathbf{m}) = \sum_{t=0}^{T} w_t \mu_t(\mathbf{m}) \tag{4.33}$$

$$\text{s.t.} \quad R(\mathbf{m}) - (g_0 - z_0^-)\mu_0(\mathbf{m}) \geqslant z_0^- \tag{4.34}$$

$$(\mu_t(\mathbf{m}) - 1)\delta_t + G_t(\mathbf{m}) \leqslant b_t; \quad t = 1,2,\cdots,T \tag{4.35}$$

$$1 \leqslant m_i \leqslant u_i; \quad i = 1,2,\cdots,N \tag{4.36}$$

其中 w_t，$t=0,1,\cdots,T$ 是由决策者给出的隶属函数 μ_t 的权重，而且 $\sum_{t=0}^{T} w_t = 1$。模糊可靠性优化问题就是要确定每个子系统最好的冗余单元数，使系统设计者对可靠性目标和资源的最好利用量的满意度达到最大。

考虑以下实例，该问题是由 Tillman 的文章[403]中的问题修改得到的。在满足模糊非线性约束的条件下，最大化非线性可靠性目标如下：

$$\max \quad R(\mathbf{m}) = \prod_{i=1}^{6} [1 - (1 - (1 - q_{i1})^{m_i+1}) - \sum_{u=2}^{4} (q_{iu})^{m_i+1}] \gtrsim 0.93$$

$$\text{s.t.} \quad G_1(\mathbf{m}) = (m_1 + 3)^2 + (m_2)^2 + (m_3)^2 + (m_4 + 3)^2 + (m_5)^2 + (m_6)^2 \lesssim 51$$

$$G_2(\mathbf{m}) = 20 \sum_{i=1}^{6} \{m_i + \exp(-m_i)\} \gtrsim 260$$

$$G_3(\mathbf{m}) = 20 \sum_{i=1}^{6} \{m_i \exp(-m_i/4)\} \gtrsim 140$$

$$1 \leqslant m_i \leqslant 3，正整数； \quad i = 1,2,\cdots,6$$

其中 $\mathbf{m} = [m_1, \cdots, m_6]$。子系统受限于一种 O 类，三种 A 类，共四种失效模式($s_i = 4$)，$i=1,\cdots,6$。对于每个子系统，失效概率如表 4.7 所示。

表 4.7　每个子系统的失效模式和失效概率

子系统 i	失效模式 $s_i=4$，$h_i=1$	失效概率 q_{iu}	子系统 i	失效模式 $s_i=4$，$h_i=1$	失效概率 q_{iv}
1	O	0.002	4	O	0.003
	A	0.05		A	0.01
	A	0.10		A	0.08
	A	0.18		A	0.10
2	O	0.004	5	O	0.002
	A	0.02		A	0.02
	A	0.15		A	0.10
	A	0.12		A	0.07
3	O	0.005	6	O	0.002
	A	0.05		A	0.02
	A	0.20		A	0.10
	A	0.10		A	0.10

令 $g_0 = 0.93, z_0^- = 0.86, \delta_1 = 14.0, \delta_2 = 8.0, \delta_3 = 8.0, w_0 = 0.85, w_1 = 0.05, w_2 = 0.05, w_3 = 0.05$，以上问题可以转化为如下等价的清晰问题：

$$\begin{aligned}
\max \quad & f(\mathbf{m}) = 0.85\,\mu_0(\mathbf{m}) + 0.05\,\mu_1(\mathbf{m}) + 0.05\,\mu_2(\mathbf{m}) + 0.05\,\mu_3(\mathbf{m}) \\
\text{s.t.} \quad & R(\mathbf{m}) - (g_0 - z_0^-)\,\mu_0(\mathbf{m}) \geqslant z_0^- \\
& (\mu_1(\mathbf{m}) - 1)\,\delta_1 + (m_1 + 3)^2 + (m_2)^2 + (m_3)^2 + (m_4 + 3)^2 \\
& \qquad + (m_5)^2 + (m_6)^2 \leqslant 51 \\
& (1 - \mu_2(\mathbf{m}))\,\delta_2 + 20\sum_{i=1}^{6}\{m_i + \exp(-m_i)\} \geqslant 260 \\
& (1 - \mu_3(\mathbf{m}))\,\delta_3 + 20\sum_{i=1}^{6}\{m_i \exp(-m_i/4)\} \geqslant 140 \\
& m_i \geqslant 0，整数； \quad i = 1,2,\cdots,6
\end{aligned}$$

4.5.2 遗传算法与计算实例

1. 染色体与初始种群

染色体定义为冗余单元数 m_{ki} 的有序表：

$$\mathbf{v}_k = [\,m_{k1} \quad m_{k2} \quad \cdots \quad m_{k6}\,]$$

其中，$\mathbf{v}_k$ 是第 k 个染色体。

染色体的初始种群在[1,3]范围内随机产生。令 $pop_size=5$，初始染色体如下：

$$\mathbf{v}_1=[1\ 2\ 3\ 3\ 1\ 3]$$
$$\mathbf{v}_2=[3\ 1\ 2\ 2\ 1\ 3]$$
$$\mathbf{v}_3=[1\ 2\ 3\ 3\ 3\ 2]$$
$$\mathbf{v}_4=[2\ 2\ 3\ 3\ 1\ 1]$$
$$\mathbf{v}_5=[3\ 1\ 2\ 2\ 3\ 2]$$

2. 染色体的评估

评估染色体时，对于每个合法的染色体，赋给目标函数值 $f(\mathbf{m})$，而对于每个非法的染色体，则给予惩罚。令其适值为 0，有

$$eval(\mathbf{v}_k)=\begin{cases}f(\mathbf{m}), & R(\mathbf{m})\geqslant z_0^-,\ G_t(\mathbf{m})\leqslant b_t+\delta_t,\ 1\leqslant m_i\leqslant u_i,\quad \forall i,t\\ 0, & \text{其他}\end{cases}\tag{4.37}$$

以上染色体的适值为

$$eval(\mathbf{v}_1)=0.271185$$
$$eval(\mathbf{v}_2)=0.408735$$
$$eval(\mathbf{v}_3)=0.416843$$
$$eval(\mathbf{v}_4)=0.532714$$
$$eval(\mathbf{v}_5)=0.555988$$

3. 交叉

利用单点交叉法。令交叉率 $p_c=0.34$，随机数序列为

0.089663　0.215613　0.345064　0.526048　0.641652

这意味选择染色体 $\mathbf{v}_1$ 和 $\mathbf{v}_2$ 进行交叉。断点的位置在[1，6]中随机产生，假定位置是 3，有

$$\mathbf{v}_1=[\,1\ 2\ 3\ |\ 3\ 1\ 3\,]$$
$$\updownarrow$$
$$\mathbf{v}_2=[\,3\ 1\ 2\ |\ 2\ 1\ 3\,]$$

相互交换双亲的右部分，得到的后代为

$$\mathbf{o}_1=[\,1\ 2\ 3\ 2\ 1\ 3\,]$$
$$\mathbf{o}_2=[\,3\ 1\ 2\ 3\ 1\ 3\,]$$

后代的适值是

$$eval(\mathbf{o}_1)=0.291535$$
$$eval(\mathbf{o}_2)=0.388195$$

4. 变异

用随机摄动作为变异。对于一个选用的基因 m_{ki}，用[1,3]的随机整数来代替。令变异率为 $p_m=0.2$，即平均有 $6\times5\times0.2=6$ 位作变异。在产生的 30 个随机数中，6 个随机

数小于 0.2。位数和相应的染色体及染色体中的位置如表 4.8 所示。

表 4.8 位数及染色体位置

位数	染色体	位置	随机数
16	3	4	0.113498
18	3	6	0.038301
29	5	5	0.196936
39	7	3	0.034211
41	7	5	0.154180
42	7	6	0.134129

变异产生的后代为

$$\mathbf{o}_3 = [\ 1\ 2\ 3\ 2\ 3\ 1\ ]$$
$$\mathbf{o}_4 = [\ 3\ 1\ 2\ 2\ 2\ 2\ ]$$
$$\mathbf{o}_5 = [\ 3\ 1\ 3\ 2\ 3\ 1\ ]$$

后代的适值为

$$eval(\mathbf{o}_3) = 0.227088$$
$$eval(\mathbf{o}_4) = 0.565843$$
$$eval(\mathbf{o}_5) = 0.375953$$

5. 选择

采用确定性的选择策略，也就是说，删除双亲和后代中所有相同的个体，并将所有个体按照降序排列，选择前 *pop_size* 个染色体组成新一代种群：

$$\mathbf{v}_1' = [\ 3\ 1\ 2\ 2\ 2\ 2\ ] \quad (\mathbf{o}_4), \quad eval(\mathbf{v}_1') = 0.565843$$
$$\mathbf{v}_2' = [\ 3\ 1\ 2\ 2\ 3\ 2\ ] \quad (\mathbf{v}_5), \quad eval(\mathbf{v}_2') = 0.555988$$
$$\mathbf{v}_3' = [\ 2\ 2\ 3\ 3\ 1\ 1\ ] \quad (\mathbf{v}_4), \quad eval(\mathbf{v}_4') = 0.532714$$
$$\mathbf{v}_4' = [\ 1\ 2\ 3\ 3\ 3\ 2\ ] \quad (\mathbf{v}_3), \quad eval(\mathbf{v}_3') = 0.416843$$
$$\mathbf{v}_5' = [\ 3\ 1\ 2\ 2\ 1\ 3\ ] \quad (\mathbf{v}_2), \quad eval(\mathbf{v}_1') = 0.408735$$

遗传算法随机运行 300 代获得的最好解是

$$\mathbf{v}^* = [\ 3\ 3\ 3\ 2\ 2\ 2\ ], \quad R(\mathbf{m}^*) = 0.928987$$

相应的隶属函数值为

$$\mu_0 = 0.986, \quad \mu_1 = 1.0, \quad \mu_2 = 1.0, \quad \mu_3 = 1.0$$

4.6 区间系数的可靠性优化

在过去的 30 年里，非确定性的决策问题有三种主要的方法：

(1) 随机规划；

(2) 模糊规划；

(3) 区间规划。

在随机规划方法中，数学模型的系数视为随机变量，并假定其概率分布是已知的。在

模糊规划方法中，系数是模糊集，其隶属函数是已知的。然而，对于现实问题的许多应用实例，决策者很难确定系数的概率分布或隶属函数。另一方面，这样的非确定性很容易用置信区间来表示，这是区间代数和区间规划发展的基础[232,311]。

Gen 等人研究了非确定型系数的系统可靠性的最优设计问题，这些系数可以用置信区间近似地表达。这样非确定型系数的系统可靠性的最优设计问题就可以用区间规划模型来描述。第二章中关于区间规划的求解方法可以用来求解此问题[150]。其基本思想是首先把非线性区间规划模型转化为等价的双目标非线性规划模型，然后用遗传算法找出 Pareto 解集。该问题有如下特点：

(1) 组合特性；

(2) 目标和约束的非线性；

(3) 多目标。

这使得问题的求解更为困难。为此，采用目标空间自适应移动线方法来构造适值函数，迫使遗传搜索寻找目标空间的非劣点，然后用自适应惩罚技术引导遗传搜索从可行和非可行两个方向来趋近 Pareto 解。

4.6.1 问题的描述

可靠性优化问题的传统描述均假定模型的系数是常量，并用确定型的优化模型来处理。系统可靠性的最优设计是在系统设计阶段就要解决的，然而在系统设计阶段模型系数是非确定的和非精确的，常常很难确定其精确值。不过这些系数可以用置信区间来近似给出，也就是说我们可以用区间规划方法来处理系统可靠性最优设计中的非确定性。

考虑 4.1.3 节中的问题。假定系统包含 N 个子系统，每个子系统有多种设计可供选择。问题是在系统的费用和重量的允许范围内，确定选哪种可选设计，用多少个冗余单元，可使系统达到最大的可靠性。这个问题常用非线性整数规划来表达，对于非确定的情形，参数用区间数来代替，问题可表达为如下区间规划模型[150]：

$$\max \quad R(\mathbf{m},\boldsymbol{\alpha}) = \prod_{i=1}^{N}(1-(1-R_i(\alpha_i))^{m_i}) \tag{4.38}$$

$$\text{s.t.} \quad G_1(\mathbf{m},\boldsymbol{\alpha}) = \sum_{i=1}^{N} C_i(\alpha_i)m_i \leqslant C \tag{4.39}$$

$$G_2(\mathbf{m},\boldsymbol{\alpha}) = \sum_{i=1}^{N} W_i(\alpha_i)m_i \leqslant W \tag{4.40}$$

$$1 \leqslant \alpha_i \leqslant \beta_i,\ \text{整数}; \quad i = 1,\cdots,N \tag{4.41}$$

$$1 \leqslant m_i \leqslant u_i,\ \text{整数}; \quad i = 1,\cdots,N \tag{4.42}$$

其中：

N——子系统总数；

m_i——子系统 i 中冗余单元的个数；

α_i——子系统 i 的可选设计；

$\mathbf{m}$——向量，$\mathbf{m}=[m_1,\ m_2,\ \cdots,\ m_N]$；

$\boldsymbol{\alpha}$——向量，$\boldsymbol{\alpha}=[\alpha_1,\ \alpha_2,\ \cdots,\ \alpha_N]$；

β_i——α_i 的上界；

u_i——m_i 的上界；

$R_i(\alpha_i)$——区间可靠性，$R_i(\alpha_i)=[r_i^L(\alpha_i),r_i^R(\alpha_i)]$；

$C_i(\alpha_i)$——区间系数，$C_i(\alpha_i)=[c_i^L(\alpha_i),c_i^R(\alpha_i)]$；

$W_i(\alpha_i)$——区间系数，$W_i(\alpha_i)=[w_i^L(\alpha_i),w_i^R(\alpha_i)]$；

C——系统费用的区间约束，$C=[c^L,c^R]$；

W——系统重量的区间约束，$W=[w^L,w^R]$。

于是，问题可以转化为等价的清晰双目标规划问题。由区间规划转化为双目标规划有两个关键步骤：

(1) 对于两个区间，利用不等式成立度的定义，把区间约束转化为等价的清晰约束；

(2) 利用区间数之间的顺序定义，把区间目标转化为等价的两个清晰目标。

首先，对等式(4.38)取自然对数，有

$$Z(\mathbf{m},\boldsymbol{\alpha})=\ln R(\mathbf{m},\boldsymbol{\alpha}) \tag{4.43}$$

$$=\sum_{i=1}^{N} D_i(m_i,\alpha_i) \tag{4.44}$$

其中 $D_i(m_i,\alpha_i)$是区间系数的目标函数，可以进一步写成

$$D_i(m_i,\alpha_i)=\ln(1-(1-R_i(\alpha_i))^{m_i}) \tag{4.45}$$

$$=[d_i^L(m_i,\alpha_i),d_i^R(m_i,\alpha_i)] \tag{4.46}$$

其中

$$d_i^L(m_i,\alpha_i)=\ln(1-(1-r_i^L(\alpha_i)^{m_i}) \tag{4.47}$$

$$d_i^R(m_i,\alpha_i)=\ln(1-(1-r_i^R(\alpha_i)^{m_i}) \tag{4.48}$$

根据 Nakahara 和 Gen[312]给出的定理 2.3，在保持区间可靠性的顺序关系不变的意义上，区间目标(4.44)等价于非线性区间目标。根据 Ishibuchi 和 Tanaka[232]给出的定理 2.2，对于给定的约束(4.39)和(4.40)不等式成立度 h_1 和 h_2 的值，问题(4.38)～(4.42)可以转化为如下形式：

$$\max \quad z^L(\mathbf{m},\boldsymbol{\alpha})=\sum_{i=1}^{N} d_i^L(m_i,\alpha_i) \tag{4.49}$$

$$\max \quad z^C(\mathbf{m},\boldsymbol{\alpha})=\sum_{i=1}^{N} \frac{d_i^L(m_i,\alpha_i)+d_i^R(m_i,\alpha_i)}{2} \tag{4.50}$$

$$\text{s.t.} \quad g_1(\mathbf{m},\boldsymbol{\alpha})=\sum_{i=1}^{N} c_i(\alpha_i)\, m_i \leqslant c \tag{4.51}$$

$$g_2(\mathbf{m},\boldsymbol{\alpha})=\sum_{i=1}^{N} w_i(\alpha_i)\, m_i \leqslant w \tag{4.52}$$

$$1\leqslant m_i\leqslant u_i,\ \text{整数}; \quad i=1,2,\cdots,N \tag{4.53}$$

$$1\leqslant \alpha_i\leqslant \beta_i,\ \text{整数}; \quad i=1,2,\cdots,N \tag{4.54}$$

其中约束中的系数由下式确定：

$$c_i(\alpha_i)=h_1\, c_i^R(\alpha_i)+(1-h_1)\, c_i^L(\alpha_i) \tag{4.55}$$

$$w_i(\alpha_i)=h_2\, w_i^R(\alpha_i)+(1-h_2)\, w_i^L(\alpha_i) \tag{4.56}$$

$$c=(1-h_1)\, c^R+h_1\, c^L \tag{4.57}$$

$$w = (1 - h_2) w^R + h_2 w^L \tag{4.58}$$

4.6.2 遗传算法

用遗传算法求解问题(4.49)~(4.54)的基本思想与第二章中区间规划的求解思想一致。

1. 染色体的表达与初始种群

染色体用下式定义：

$$\mathbf{v}_k = [\ (\alpha_{k1},\ m_{k1}),\ (\alpha_{k2},\ m_{k2}),\ \cdots,\ (\alpha_{kN},\ m_{kN})\ ]$$

其中，下标 k 是染色体的标号；α_{ki}是子系统 i 的备选设计；m_{ki}是冗余单元。

初始种群在所有的 $m_{ki} \in [1, u_i]$和 $\alpha_{ki} \in [1, \beta_i]$的条件下随机产生。

2. 交叉与变异

用均匀交叉算子作交叉，在整数变量允许的范围内采用随机摄动实现变异。

3. 选择

采用确定性选择，即删除所有与双亲和后代中相同的个体，按照降序排列，选择前 *pop_size* 个染色体组成新一代种群。

4. 评估

在评估阶段，有两项重要的工作，即如何处理非可行的染色体和根据双目标来确定染色体的适值。

令 $\mathbf{v}_k$ 是当前代中的第 k 个染色体，评估函数定义为

$$eval(\mathbf{v}_k) = (w_1 z^L(\mathbf{m}_k, \boldsymbol{\alpha}_k) + w_2 z^C(\mathbf{m}_k, \boldsymbol{\alpha}_k))\ p(\mathbf{m}_k, \boldsymbol{\alpha}_k) \tag{4.59}$$

评估函数包括两项，加权目标项和惩罚项。加权目标项主要用于给出选择压力，迫使遗传搜索趋近于 Pareto 解，惩罚项则使遗传搜索从可行域和非可行域两侧趋于 Pareto 解。

5. 加权和目标的计算

令 E 表示目前为止检查到的非全优解的集合。E 中有两个特别的点是我们最感兴趣的，其一包含 $z^L(\mathbf{m}_k, \boldsymbol{\alpha}_k)$的最大值，另一个包含 $z^C(\mathbf{m}_k, \boldsymbol{\alpha}_k)$的最大值。这两点分别表示为$(z^C_{\min}, z^L_{\max})$和$(z^C_{\max},\ z^L_{\min})$，其中

$$z^C_{\min} = \min\ \{z^C(\mathbf{m}_k, \boldsymbol{\alpha}_k) \mid \mathbf{m}_k, \boldsymbol{\alpha}_k \in E\}$$
$$z^C_{\max} = \max\ \{z^C(\mathbf{m}_k, \boldsymbol{\alpha}_k) \mid \mathbf{m}_k, \boldsymbol{\alpha}_k \in E\}$$
$$z^L_{\min} = \min\ \{z^L(\mathbf{m}_k, \boldsymbol{\alpha}_k) \mid \mathbf{m}_k, \boldsymbol{\alpha}_k \in E\}$$
$$z^L_{\max} = \max\ \{z^L(\mathbf{m}_k, \boldsymbol{\alpha}_k) \mid \mathbf{m}_k, \boldsymbol{\alpha}_k \in E\}$$

加权目标函数可以按下式构造：

$$w_1 z^L(\mathbf{m}_k, \boldsymbol{\alpha}_k) + w_2 z^C(\mathbf{m}_k, \boldsymbol{\alpha}_k)$$

其中

$$w_1 = z^C_{\max} - z^C_{\min}$$

$$w_2 = z_{max}^L - z_{min}^L$$

由点(z_{min}^C, z_{max}^L)和(z_{max}^C, z_{min}^L)构成的直线将目标空间分成两部分：一部分包含正的理想解，用Z^+表示，另一部分包含负的理想解，用Z^-表示。可行解空间F也相应地分成两部分：一部分是$F^- = F \cap Z^-$，另一部分是$F^+ = F \cap Z^+$。很容易验证F^+中的解比F^-中的解适值大。因此空间F^+中的染色体有相对多的机会进入下一代。在每一代中，Pareto集E不断更新，相应地两个特别点也得到更新。这意味着，随着进化过程的进行，由这两点构成的线逐渐地沿着从负的理想解向正的理想解方向移动（称为自适应移动线）。也就是说，适值函数产生选择压力迫使遗传搜索在搜索空间中向非全优点靠近。

6. 惩罚

惩罚函数由下式构造：

$$p(\mathbf{m}_k, \boldsymbol{\alpha}_k) = 1 - \frac{1}{2}\sum_{i=1}^{2}\left(\frac{\Delta b_i(\mathbf{m}_k, \boldsymbol{\alpha}_k)}{\Delta b_i^{max}}\right) \tag{4.60}$$

式中

$$\Delta b_i(\mathbf{m}_k, \boldsymbol{\alpha}_k) = \max\{0, g_i(\mathbf{m}_k, \boldsymbol{\alpha}_k) - b_i\} \tag{4.61}$$

$$\Delta b_i^{max} = \max\{\varepsilon, \Delta b_i(\mathbf{m}_k, \boldsymbol{\alpha}_k); k = 1, \cdots, pop_size\} \tag{4.62}$$

其中，$\Delta b_i(\mathbf{m}_k, \boldsymbol{\alpha}_k)$是第$k$个染色体违背约束$j$的值；$\Delta b_i^{max}$是当前代中约束$i$的最大违背值；$\varepsilon$是用于惩罚避免除零的最小正数。惩罚项促使遗传搜索从可行域和非可行域两侧达到Pareto解。

7. 总体步骤

遗传算法过程

```
begin
  t←0;
  初始化 P(t);
  形成 Pareto 解集 E(t);
  评估 P(t);
  while 不满足终止条件 do
    重组 P(t);
    更新 Pareto 解集 E(t);
    评估 P(t);
    从 P(t)中选择 P(t+1);
    t←t+1;
  end
end
```

其中，更新集合$E(t)$如下：

更新集合E的过程

```
begin
  对于每个染色体，计算双目标函数值；
```

增加 E 中新的非全优点；

删除 E 中劣势点；

确定新的特别点 $(z^C_{\min}, z^L_{\max})$ 和 $(z^C_{\max}, z^L_{\min})$；

end

4.6.3 计算实例

本实例是文献[145]中问题的扩展。系统包含 14 个子系统，每个子系统有 3 或 4 个备选设计，可能最大的冗余单元数为 6，问题的区间系数如表 4.9 所示。

表 4.9 测试问题的区间系数

子系统 i	备选设计号					
	1			2		
	r_i^L, r_i^R	c_i^L, c_i^R	w_i^L, w_i^R	r_i^L, r_i^R	c_i^L, c_i^R	w_i^L, w_i^R
1	[0.86,0.91]	[1,4]	[3,6]	[0.88,0.94]	[1,4]	[3,8]
2	[0.92,0.97]	[2,6]	[7,11]	[0.89,0.94]	[1,3]	[10,13]
3	[0.80,0.87]	[1,3]	[4,9]	[0.85,0.91]	[3,6]	[5,9]
4	[0.78,0.85]	[1,5]	[4,6]	[0.82,0.88]	[4,7]	[6,9]
5	[0.89,0.94]	[2,5]	[4,8]	[0.88,0.93]	[2,5]	[3,7]
6	[0.93,0.99]	[3,9]	[5,10]	[0.93,0.99]	[5,9]	[3,8]
7	[0.86,0.92]	[4,7]	[7,10]	[0.87,0.93]	[4,8]	[8,11]
8	[0.76,0.82]	[3,6]	[4,9]	[0.85,0.91]	[5,9]	[7,10]
9	[0.92,0.98]	[2,7]	[8,12]	[0.93,0.99]	[3,8]	[9,14]
10	[0.78,0.84]	[4,9]	[6,10]	[0.80,0.86]	[4,7]	[5,8]
11	[0.89,0.95]	[3,7]	[5,9]	[0.90,0.95]	[4,9]	[6,10]
12	[0.74,0.80]	[1,4]	[4,9]	[0.77,0.83]	[3,6]	[5,8]
13	[0.93,0.99]	[2,6]	[5,10]	[0.94,0.98]	[3,8]	[5,10]
14	[0.85,0.91]	[4,7]	[6,10]	[0.87,0.92]	[4,7]	[7,10]

子系统 i	备选设计号					
	3			4		
	r_i^L, r_i^R	c_i^L, c_i^R	w_i^L, w_i^R	r_i^L, r_i^R	c_i^L, c_i^R	w_i^L, w_i^R
1	[0.87,0.93]	[2,6]	[2,6]	[0.90,0.96]	[4,8]	[5,9]
2	[0.88,0.94]	[2,5]	[9,13]	—	—	—
3	[0.82,0.88]	[1,4]	[6,10]	[0.87,0.93]	[4,7]	[4,8]
4	[0.80,0.86]	[5,9]	[4,9]	—	—	—
5	[0.90,0.96]	[3,8]	[5,9]	—	—	—
6	[0.92,0.98]	[2,6]	[5,10]	[0.91,0.97]	[2,7]	[4,8]
7	[0.89,0.95]	[5,9]	[9,13]	—	—	—
8	[0.86,0.92]	[6,9]	[6,10]	—	—	—
9	[0.91,0.96]	[4,7]	[7,11]	[0.86,0.92]	[3,7]	[8,12]
10	[0.85,0.90]	[4,8]	[4,7]	—	—	—
11	[0.91,0.96]	[5,10]	[6,10]	—	—	—
12	[0.80,0.86]	[4,7]	[6,9]	[0.85,0.91]	[5,8]	[7,12]
13	[0.92,0.97]	[3,7]	[6,10]	—	—	—
14	[0.90,0.95]	[5,10]	[5,10]	[0.94,0.99]	[6,11]	[9,14]

注：$C=[95,130]$，$W=[163,177]$，$u_i=5$，$i=1,\cdots,14$，

$\beta_i=4$，$i=1,3,6,9,12,14$ 和 $\beta_i=3$，$i=2,4,5,7,8,10,11,13$。

遗传算法的参数设置为：种群大小 $pop_size=40$，交叉率 $p_c=0.4$，变异率 $p_m=0.4$，最大代数 $max_gen=2\ 000$. 并设定 $h_1=0.5$，$h_2=0.5$.

用以上算法求得的 Pareto 解见表 4.10。相应的最优设计(解)见表 4.11。解空间中这些解见图 4.8，相应的区间数(区间目标)见图 4.8. 从图中可以看出，这些解在顺序关系 $\leqslant_{LC}$下是不可比的。

表 4.10 Pareto 解

n	z^L	z^R
1	0.97801	0.98758
2	0.97826	0.98660
3	0.97832	0.98647

表 4.11 系统可靠性的最优设计

Pareto 解	1		2		3	
子系统	设计	单元	设计	单元	设计	单元
1	1	5	3	5	1	5
2	1	5	1	5	1	5
3	1	4	2	4	1	4
4	1	5	1	4	1	5
5	2	3	1	3	2	3
6	3	3	2	3	3	3
7	2	5	2	3	2	5
8	1	4	1	5	2	3
9	2	4	1	2	2	4
10	1	4	1	5	1	4
11	2	2	1	5	2	2
12	3	4	2	4	3	4
13	1	4	1	2	1	4
14	3	4	3	5	3	4

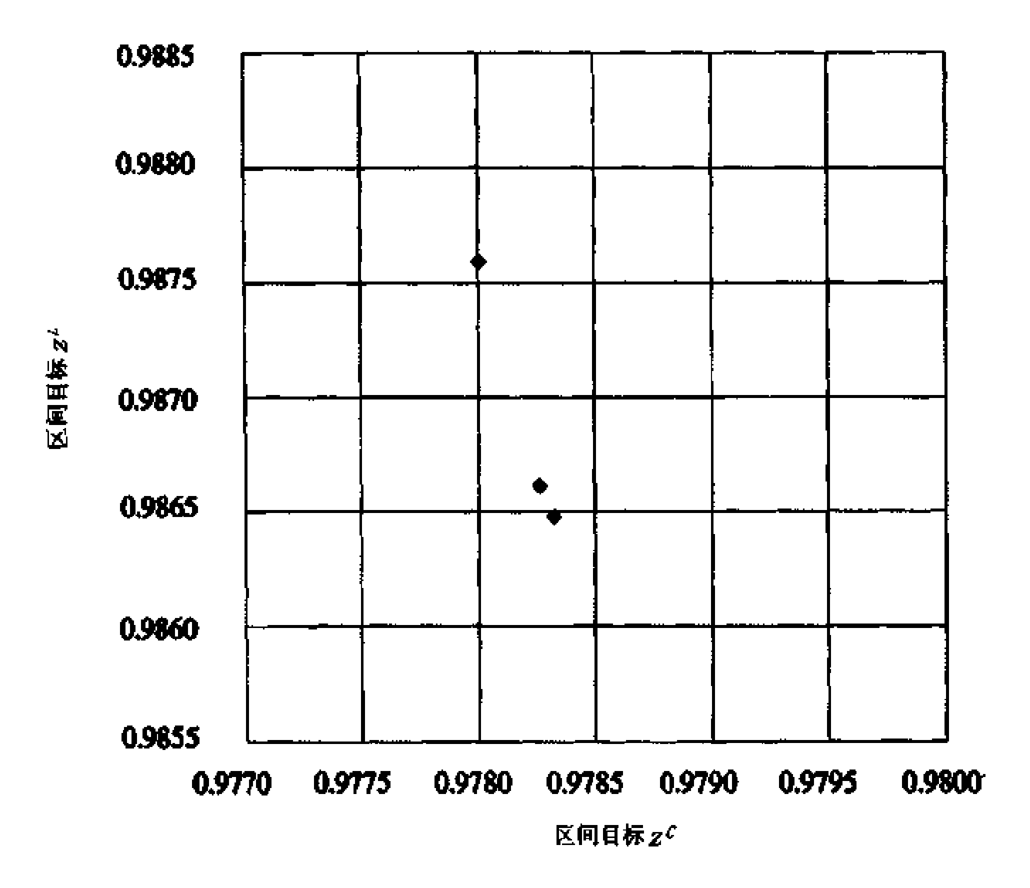


图 4.7　目标空间的 Pareto 解

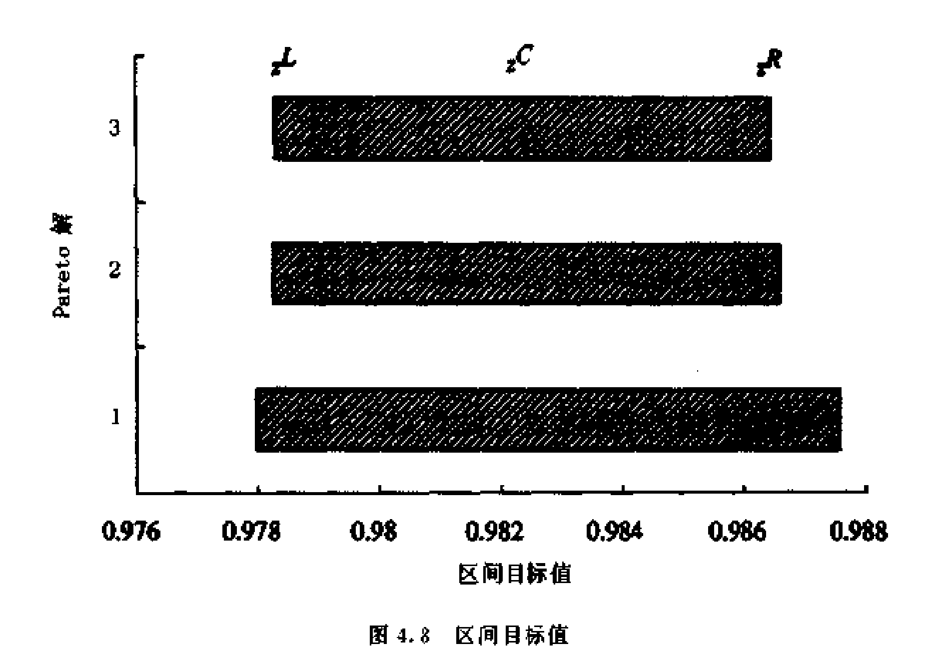


图 4.8　区间目标值

第五章　流水车间调度问题

5.1　引　　言

自从 Johnson 1954 年发表第一篇关于流水车间调度问题的文章[236]以来，流水车间调度问题引起了许多学者的关注。流水车间调度问题一般可以描述为 n 个工件要在 m 台机器上加工，每个工件需要经过 m 道工序，每道工序要求不同的机器。n 个工件在 m 台机器上的加工顺序相同。工件 i 在机器 j 上的加工时间是给定的，设为 $t_{ij}(i=1,\cdots,n;j=1,\cdots,m)$。问题的目标是求 n 个工件在每台机器上最优的加工顺序，使最大流程时间达到最小。对该问题常常作如下假设：

(1) 每个工件在机器上的加工顺序是 $1,2,\cdots,m$；

(2) 每台机器同时只能加工一个工件；

(3) 一个工件不能同时在不同的机器上加工；

(4) 工序不能预订；

(5) 工序的准备时间与顺序无关，且包含在加工时间中；

(6) 工件在每台机器上的加工顺序相同，且是确定的。

流水车间调度问题可以分为

(1) 确定型流水车间问题；

(2) 随机型流水车间问题；

(3) 模糊型流水车间问题。

在确定型流水车间问题，假定工件的加工时间是已知的确定量，在随机型流水车间问题，加工时间按照一定的概率分布而变化[115]；在模糊型决策情况下，每个工件的模糊交货期表示为决策者对工件完工时间的满意度[233,405]。此外，如果某一给定的工件在一台或多台机器上的加工时间为 0，则流水车间问题称为广义流水车间问题，否则称为纯流水车间问题。在过去的 30 年里，多数研究工作集中于纯的确定型流水车间调度问题，常用 $n/m/F/c_{\max}$ 表示，即 n 个工件/m 台机器/流水车间/最大流程时间。

令 $c(j_i,k)$ 表示工件 j_i 在机器 k 上的加工完工时间，$\{j_1,j_2,\cdots,j_n\}$ 表示工件的调度，那么 n 个工件 m 台机器的流水车间问题的工件完工时间是

$$c(j_1,1)=t_{j_11} \tag{5.1}$$

$$c(j_1,k)=c(j_1,k-1)+t_{j_1k};\quad k=2,\cdots,m \tag{5.2}$$

$$c(j_i,1)=c(j_{i-1},1)+t_{j_i1};\quad i=2,\cdots,n \tag{5.3}$$

$$c(j_i,k)=\max\{c(j_{i-1},k);c(j_i,k-1)\}+t_{j_ik};\quad i=2,\cdots,n;k=2,\cdots,m \tag{5.4}$$

最大流程时间为

$$c_{\max}=c(j_n,m) \tag{5.5}$$

一般地，常用甘特图表达流水车间问题的调度。例如，4 个工件 2 台机器问题的甘特图如

图 5.1 所示。

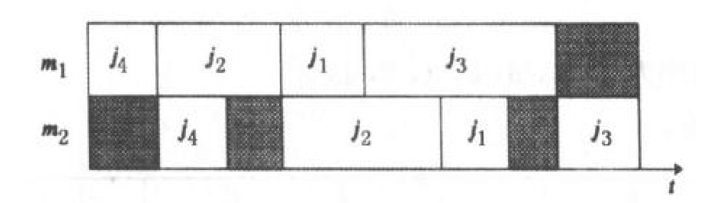


图 5.1 流水车间调度问题的甘特图

从文献上看，近年来对流水车间问题做了深入的研究，综述文章可参见文献[115]；有关著作可参见文献[299]和[41]。

5.2 两台机器的流水车间问题

对于以最大流程时间为目标的两台机器流水车间问题称为 Johnson 问题，其最优调度由著名的 Johnson 规则[21]确定。Johnson 规则的基本思想为后来 m 台机器问题的启发式算法提供了基础。

假定有 n 个工件，在第一台机器和第二台机器上的加工时间分别为 t_{i1}，$i = 1,2,\cdots,n$ 和 t_{i2}，$i = 1,2,\cdots,n$，最优调度由如下规则给出：

定理 5.1（Johnson 规则） 最优调度中工件 i 排在工件 j 之前，如果 $\min\{t_{i1}, t_{j2}\} \leqslant \min\{t_{i2}, t_{j1}\}$。

最优顺序可以直接利用这个结果通过两个工序之间的检查来构造。令 J 表示工件序列，S 表示顺序，Johnson 算法可以描述为

算法：Johnson 规则

步骤 1：令 $U = \{j \mid t_{j1} < t_{j2}\}$，$V = \{j \mid t_{j1} \geqslant t_{j2}\}$；

步骤 2：对 U 中的工件按 t_{j1} 非减顺序调度；

步骤 3：对 V 中的工件按 t_{j2} 非增顺序调度；

步骤 4：有序集 U 放到 V 之前构成工件的最优加工顺序。

为了说明本算法，考虑表 5.1 所示的 8 个工件问题。

表 5.1 8 个工件问题的实例

工件 i	1	2	3	4	5	6	7	8
t_{i1}	5	2	1	7	6	3	7	5
t_{i2}	2	6	2	5	6	7	2	1

本例的解构造如下：

步骤 1：工件集 $U = \{2, 3, 6\}$ 和 $V = \{1, 4, 5, 7, 8\}$；

步骤 2：U 中的工件调度为

工件 i：3 2 6

t_{i1}：1 2 3

步骤 3：V 中的工件调度为

$$\begin{array}{llllll} \text{工件}\ i: & 5 & 4 & 7 & 1 & 8 \\ t_{i2}: & 6 & 5 & 2 & 2 & 1 \end{array}$$

步骤 4：最优顺序为{3，2，6，5，4，7，1，8}.
甘特图如图 5.2 所示。

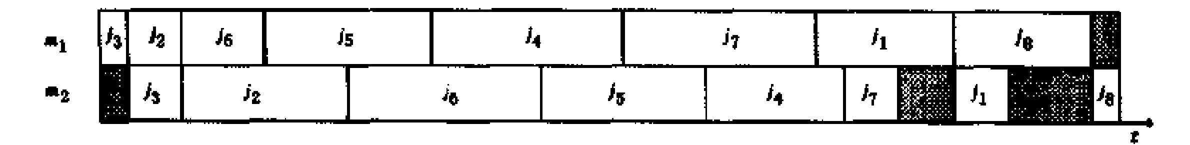


图 5.2 最优调度的甘特图

5.3 一般 m 台机器流水车间问题的启发式方法

在过去的 30 年，许多学者对纯流水车间问题作了深入的研究，但还没有一个求解最优解的简明算法。整数规划和分枝定界技术是寻求最优解的常用方法[227]，然而对于一些大规模甚至中等规模的问题，整数规划和分枝定界技术仍然不很有效。另一方面，已经证明流水车间问题是 NP 完全问题[78,356]。因此，许多学者相继提出了一些既好又快的启发式算法。下面讨论一些知名的启发式算法。

5.3.1 Palmer 启发式算法

Palmer 提出了基于工件的加工时间按斜度顺序指标(slope order index)排列工件的启发式算法，其基本思想是给每个工件赋优先权数。按机器的顺序，加工时间趋于增加的工件得到较大的优先数；与此相反，按机器的顺序，加工时间趋于减小的工件得到较小的优先数[330]。工件 i 的斜度指标(slope index)s_i 按下式计算：

$$s_i = \sum_{j=1}^{m}(2j - m - 1)\,t_{ij};\quad i = 1, 2, \cdots, n \tag{5.6}$$

按 s_i 非增的顺序排列工件，可以构造如下的工件加工序列：

$$s_{i_1} \geqslant s_{i_2} \geqslant \cdots \geqslant s_{i_n} \tag{5.7}$$

5.3.2 Gupta 启发式算法

Gupta 提出了另一种类似 Palmer 方法的启发式算法。他考虑了对于三台机器问题的 Johnson 规则的最优性这一很有趣的问题[195]，并用不同的方式定义斜度指标。按照他的定义，工件 i 的斜度指标 s_i 定义为

$$s_i = \frac{e_i}{\min\limits_{1\leqslant k\leqslant m-1}\{t_{ik} + t_{i,k+1}\}} \tag{5.8}$$

其中

$$e_i = \begin{cases} 1, & t_{i1} < t_{im} \\ -1, & t_{i1} \geqslant t_{im} \end{cases} \tag{5.9}$$

然后，根据斜度指标(5.8)排列工件的加工顺序。

5.3.3 CDS 启发式算法

Campbell，Dudek 和 Simth 启发式算法（简称 CDS 启发式算法）是 Johnson 算法的扩展[53]，其有效性依赖于两个性质：

(1) 按启发式方式使用 Johnson 规则；

(2) 一般地产生多个顺序，选其中最好解作为工件加工顺序。

算法首先把 m 台机器系统地组成两组，产生 $m-1$ 个两台机器问题的集合，然后利用 Johnson 的两台机器算法得到 $m-1$ 个加工顺序，最后选取其中最好的一个作为近优解。第一阶段，考虑由机器 1 和 m 形成的两台机器问题；第二阶段，考虑模拟的两台机器问题：模拟机器 1 由机器组 {1，2} 组成，模拟机器 2 由机器组 $\{m, m-1\}$ 组成；第 k 阶段，考虑模拟的两台机器问题：模拟机器 1 由机器组 $\{1, 2, \cdots, k\}$ 组成，模拟机器 2 由机器组 $\{m, \cdots, m-k+1\}$ 组成。组合加工时间如表 5.2 所示。

表 5.2 模拟两台机器问题的集合

阶段 i	模拟两台机器的问题		组合加工时间	
	组 1	组 2	t'_{i1}	t'_{i2}
1	1	m	t_{i1}	t_{im}
2	1,2	$m, m-1$	$t_{i1}+t_{i2}$	$t_{im}+t_{i,m-1}$
3	1,2,3	$m, m-1, m-2$	$t_{i1}+t_{i2}+t_{i3}$	$t_{im}+t_{i,m-1}+t_{i,m-2}$
…				
$m-1$	$1,2,\cdots,m-1$	$m, m-1, \cdots, 2$	$t_{i1}+t_{i2}+\cdots+t_{i,m-1}$	$t_{im}+t_{i,m-1}+\cdots+t_{i,2}$

阶段 k 的组合加工时间定义为

$$t'_{i1} = \sum_{j=1}^{k} t_{ij}, \qquad t'_{i2} = \sum_{j=1}^{k} t_{i,m-j+1} \tag{5.10}$$

CDS 启发式算法被认为是好的且具有鲁棒性的启发式，已经成为很多研究中的比较标准。

5.3.4 RA 启发式算法

Dannenbring 将 Palmer 斜度指标法和 CDS 方法结合起来，提出了一个称为快速进入(Rapid Access 简称 RA)的启发式方法[91]。其出发点是提供尽可能快捷简便的足够好的解。RA 启发式方法不需求解 $m-1$ 个模拟的两台机器问题，而仅用 Johnson 规则求解一个模拟问题，工件的加工时间按下式确定：

$$t'_{i1} = \sum_{j=1}^{m} w_{j1} t_{ij}, \qquad t'_{i2} = \sum_{j=1}^{m} w_{j2} t_{ij} \tag{5.11}$$

其中，权重定义为

$$W_1 = \{w_{j1} \mid j = 1,2,\cdots,m\} = \{m, m-1, \cdots, 2, 1\}$$

$$W_2 = \{w_{j2} \mid j = 1,2,\cdots,m\} = \{1, 2, \cdots, m-1, m\}$$

5.3.5 NEH 启发式算法

Nawaz，Enscore 和 Ham 启发式算法（简称 NEH 启发式算法）假定在所有机器上的

总加工时间越大的工件比总加工时间小的工件应该得到越大的优先数[315]。NEH 算法不把原来的 m 台机器问题转化为一个模拟的两台机器问题，而是通过每一步加入一个新工件，从而求得最好的局部解，最后构造工件的加工顺序。

NEH 启发式算法

步骤 1：按工件在机器上的总加工时间递减的顺序排列 n 个工件；

步骤 2：取前两个工件调度使部分最大流程时间达到极小；

步骤 3：从 $k=3$ 到 n 把第 k 个工件插入到 k 个可能的位置，求得最小部分的最大流程时间。

5.4 Gen-Tsujimura-Kubota 方法

遗传算法已被成功地用于求解流水车间问题[57,233,256,350,366,405]。本节详细介绍解流水车间问题的 Gen-Tsujimura-Kubota 方法。

5.4.1 表达方法

由于流水车间问题本质上是调度问题，我们可以用工件的顺序来表示染色体，这是调度问题的自然表达方法。例如，令第 k 个染色体为

$$v_k = [\ 3\ 2\ 4\ 1\ ]$$

表示工件的加工顺序是 j_3、j_2，j_4，j_1。

5.4.2 评估函数

确定每个染色体适值的简单方法是用最大流程时间的倒数。令 c_{max}^k 表示 k 个染色体的最大流程时间，那么适值为

$$eval(v_k) = 1/c_{max}^k \tag{5.12}$$

5.4.3 交叉与变异

有很多著名的交叉算子适用于顺序表达法，例如部分映射交叉(PMX)，顺序交叉(OX)和循环交叉(CX)，参见第三章。本节用 Goldberg 提出的 PMX 交叉算子，下面是给出的一个交叉实例(见图 5.3)：

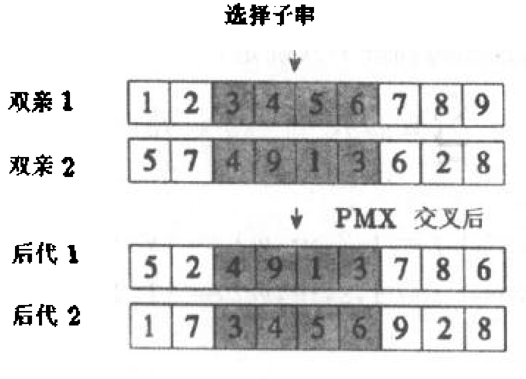


图 5.3 交叉实例

用随机交换作为变异，即随机选取一个染色体的两个基因，交换其位置，如图 5.4 所示。

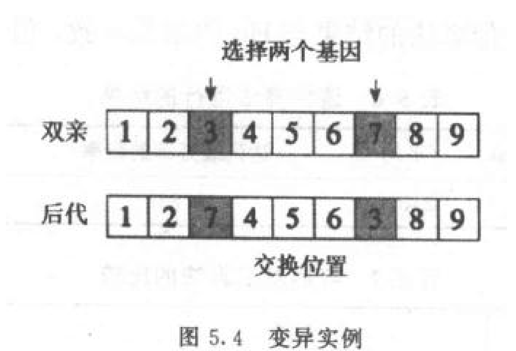


图 5.4 变异实例

5.4.4 实例

例 5.1 为简单起见，考虑由 Baker[21]给出的 5 个工件、2 台机器的问题，每个工件的加工时间如表 5.3 所示。

表 5.3 加工时间

工件 j	1	2	3	4	5
t_{j1}	3	5	1	6	7
t_{j2}	6	2	2	6	5

遗传算法(GA)的有关参数设置为：$pop_size=20$，$max_gen=20$，$p_c=0.3$，$p_m=0.1$。用遗传算法运行 10 次均得到了最优解，而且还找到了具有最优的最大流程时间的两个新的工件加工顺序。结果如表 5.4 所示，其甘特图如图 5.5 所示。

表 5.4 遗传算法与 Johnson 规则的结果比较

方法	顺序	最长完工时间	方法	顺序	最长完工时间
Johnson	3-1-4-5-2	24	GA	1-3-4-5-2	24
GA	3-1-4-5-2	24	GA	1-4-3-5-2	24

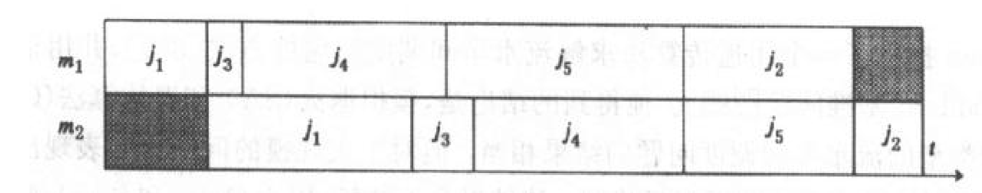


图 5.5 最优调度的甘特图

例 5.2 考虑由 Ho 和 Chang 给出的 5 个工件、4 台机器的问题[218]，加工时间如表 5.5所示：

表 5.5 加工时间

工件 j	1	2	3	4	5
t_{j1}	31	19	23	13	33
t_{j2}	41	55	42	22	5
t_{j3}	25	3	27	14	57
t_{j4}	30	34	6	13	19

参数设置：$pop_size=20$，$max_gen=150$，$p_c=0.3$，$p_m=0.1$。用遗传算法运行 12 次，其结果如表 5.6 所示，与启发式算法的结果比较如表 5.7 所示，最好解的甘特图如图 5.6 所示。比较结果说明遗传算法的结果与 Ho 的结果一致，但优于其他的启发式算法。

表 5.6　遗传算法运行的结果

总运行次数	最好解	最坏解	平均	达到最好解的频率	达到最好解的平均代数
12	213	216	213.5	0.833	37.75

表 5.7　与启发式算法的比较

方法	解	最大流程时间
GA	4-2-5-1-3	213
Ho	4-2-5-1-3	213
CDS	4-2-1-5-3	246
Dannenbring	5-1-2-3-4	256
Gupta	2-1-5-3-4	251
Palmer	5-2-4-1-3	245
随机	1-2-3-4-5	286

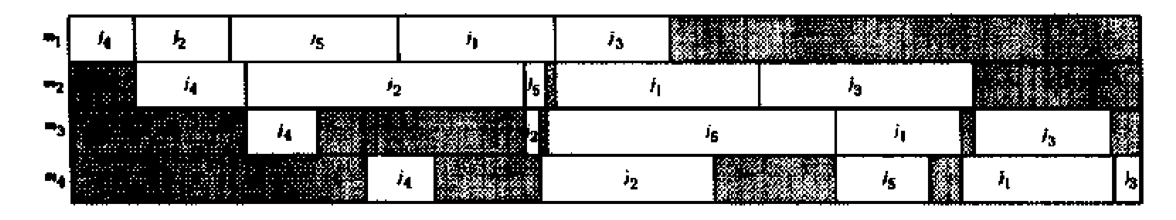


图 5.6　最优调度的甘特图

5.5　Reeves 方法

Reeves 提出了一个用遗传算法求解流水车间调度问题的方法[350]，并用他的方法检查了 Taillard 基准问题[393]。他得到的结论是，模拟退火(SA) 和遗传算法(GA)对不同规模和类型的流水车间调度问题的结果相当，但对于大规模的问题 GA 表现出更好的性能，能以更快的速度达到近似最优解。该结果令人鼓舞，因为 Ogbu 和 Smith 曾认为模拟退火比所有其他的启发式算法更优越[319]。

5.5.1　初始种群

在产生初始种群时，Reeves 结合传统的启发式算法给出了一个好的种子染色体。他用 NEH 启发式算法产生一个染色体，而其余的染色体则随机产生。比较混合式方法和没有种子的种群，他认为有种子的种群能较快地达到最后的解，并且解的质量没有降低。

Chen，Vempati 和 Aljaber 提出了几乎类似的混合式方法来产生初始种群[57]。对于 n 个工件 m 台机器流水车间问题，用 CDS 产生 $m-1$ 个调度，用 RA 启发式产生一个调度，其余的调度则由已经产生的调度随机变异产生。

5.5.2 遗传算子

Reeves 采用单点交叉，其基本思想是首先随机选取断点，然后选取第一个双亲的断点前部分作为后代的一部分，再从第二个双亲中按顺序选取合法基因填充余下部分。如图 5.7 所示：

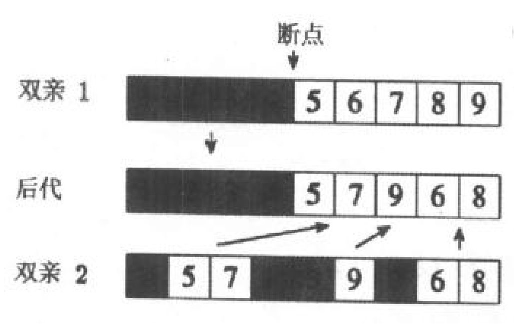


图 5.7 单点交叉

采用两种变异方法。一是互换变异，即随机选取染色体两个基因进行简单互换。另一种变异是移动变异，即随机选取一个基因，向左或右移动一个随机数位。有些实验表明移动变异似乎比互换变异好，如图 5.8 所示。

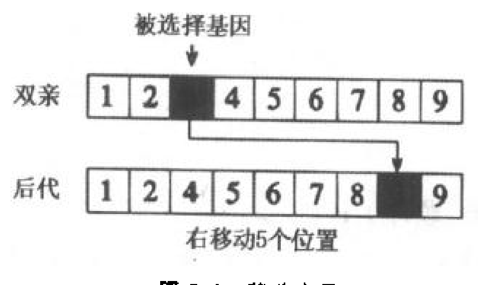


图 5.8 移动变异

5.5.3 选择

实质上，如果种群包含很多适值接近的染色体，好解与坏解的适值就没有明显的区别，在这种情况下相对适应性的度量便不能很好地区分染色体的好坏。Reeves 采用了一种简单的调度方法来作为选择的机理，即依据如下的概率分布选择双亲：

$$p_k = \frac{2k}{M(M+1)} \tag{5.13}$$

其中，k 是指按递增的顺序排列的第 k 个染色体（即最大流程时间的递减顺序）；M 是指适应性最好的染色体，这意味着中间值的染色体有 $1/M$ 的机会被选取，而最好的染色体（第 M 个染色体）有 $2/(M+1)$ 的机会，近似 2 倍中间值的机会。

Reeves 方法的伪码描述如下：

步骤 1：系统参数

设定 $M=30$，$p_c=1.0$，$p_m^{\text{init}}=0.8$，$\theta=0.99$，$p_m=p_m^{\text{init}}$，$D=0.95$。

步骤 2：初始种群

产生（NEH_顺序）；

评估（NEH_顺序）；

```
pop_no←1;
repeat
  pop_no←pop_no + 1;
  产生(随机_顺序);
  评估(随机_顺序);
until pop_no←M
调度(种群);
计算(种群_统计量)。
步骤 3: 交叉
if 随机数< p_c then
  用适应性调度分布选择(双亲_1);
  用均匀分布选择(双亲_2);
  选择(交叉点);
  交叉;
end
步骤 4: 变异
if 随机数< p_m then
  变异;
步骤 5: 评估与选择
评估(new_顺序);
在不好的成员中选择(老_顺序);
从种群中删除(老_顺序);
在种群中插入(新_顺序);
更新(种群_统计量);
步骤 6: 更新数据
p_m←θp_m;
if v_mim/v_mean> D then
p_m←p_m^init;
if cpu_time > max_cpu_time then
   停止;
else
   返回步骤 3。
end
```

5.5.4 计算实例

Taillard[393]提出了一组特别困难的测试问题，虽然用冗长的禁忌搜索(TS)步骤能够找到最好解，但仍相当地劣于它们的下界。测试题的规模从 20 个工件、5 台机器到 500 个工件、20 台机器，每个规模的问题有 10 个，工件的加工时间是 $U(1,100)$均匀分布产生的随机数。

Reeves 对这些问题的测试结果如表 5.8 所示，其中解的性能用偏离 Tailland 解的上界的平均百分数度量，这里的平均是指在每组问题的 10 个实例上取平均值。

表 5.8　遗传算法、模拟退火和邻域搜索解的质量比较

问题规模	方　法		
	(GA)	(SA)	(NSA)
20/5	1.61	1.27	1.46
20/10	2.29	1.71	2.02
20/20	1.95	0.86	1.10
50/5	0.45	0.78	0.79
50/10	2.28	1.98	3.21
50/20	3.44	2.86	3.90
100/5	0.23	0.56	0.76
100/10	1.25	1.33	2.69
100/20	2.95	2.32	3.98
200/10	0.50	0.83	3.81
200/20	1.35	1.74	6.07
500/20	−0.22	0.85	9.07
平均	1.50	1.42	3.24

从表中可以看出，除非小规模的问题，遗传算法(GA)和模拟退火(SA)要优于邻域搜索(NSA)；如果直接比较 SA 和 GA，从总体上或它们与上界的距离上来看，很难区分谁优谁劣。总的来看，SA 略微好于 GA，但对于大规模问题，实验结果的确表明 GA 性能更好一些。值得指出的是对于最后一组问题，遗传算法 10 个中有 9 个实际上求得了优于 Taillard 上界的解。

5.6　Ishibuchi-Yamamoto-Murata-Tanaka 方法

Ishibuchi，Yamamoto，Murata 和 Tanaka 提出了一个模糊流水车间调度问题，并利用遗传算法和邻域搜索算法求解了该问题[233]，结果表明遗传算法与邻域搜索的混合式方法很有效。

5.6.1　模糊流水车间问题

Ishibuchi 等人用模糊交货期的概念描述了两个模糊流水车间问题。一是对于给定工件的最小满意度的最大化问题，另一个问题是总体满意度的最大化问题。每个工件模糊交货期的隶属函数代表了决策者对工件完成时间的满意度，可以用图 5.9 所示的梯形隶属函数来表示。其中，c_j 是工件 j 的完工时间；$\mu_j(c_j)$是工件 j 的模糊交货期的隶属函数，其隶属函数可以写成：

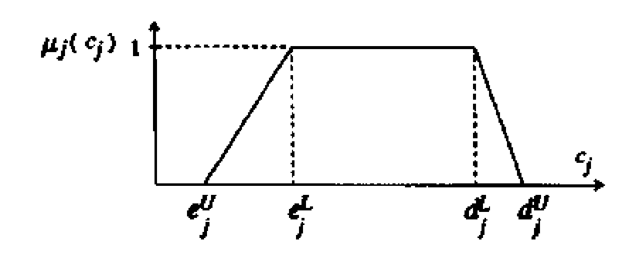


图 5.9 梯形隶属函数

$$\mu_j(c_j) = \begin{cases} 0, & c_j \leqslant e_j^U \\ 1 - (e_j^U - c_j)/(e_j^U - e_j^L), & e_j^L < c_j < e_j^U \\ 1, & e_j^U \leqslant c_j \leqslant d_j^L \\ 1 - (c_j - d_j^L)/(d_j^U - d_j^L), & d_j^L < c_j < d_j^U \\ 0, & d_j^U \leqslant c_j \end{cases} \tag{5.14}$$

那么，最小满意度的最大化问题可以描述为

$$\max f_{\min} = \min \{\mu_j(c_j);\ j = 1,2,\cdots,n\} \tag{5.15}$$

总体满意度的最大化问题描述为

$$\max f_{\text{sum}} = \sum_{j=1}^{n} \mu_j(c_j) \tag{5.16}$$

5.6.2 混合式遗传算法

1. 遗传算子

有多种遗传算子可以适用于调度问题。由于其良好的计算性能，本算法采用了基于位置的交叉和移动变异。基于位置的交叉见文献[392]，随机选择位置进行交叉，然后选取第一个双亲的位置的基因作为后代的一部分，再从第二个双亲按基因的顺序填补在后代的其他部分，构成后代，图解如下：

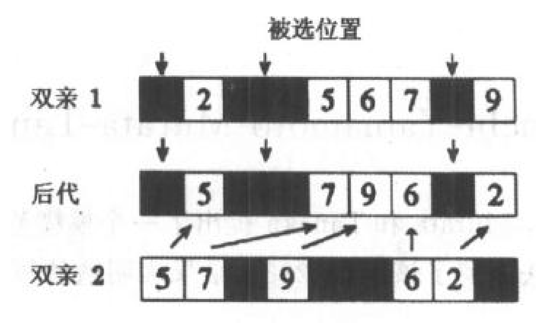


图 5.10 基于位置的交叉

2. 计算适值

令 Ψ_t 表示 第 t 代的种群，x_t^i 表示 Ψ_t 中的第 i 个染色体，即

$$\Psi_t = \{x_t^i \mid i = 1,\ 2,\ \cdots,\ pop_size\}$$

用目标函数 $f(x)$作为适值，第 t 代 Ψ_t 中第 i 个染色体的选择概率 $p(x_t^i)$按下式计算：

$$p(x_i^t) = \frac{[f(x_i^t) - f_M(\Psi_t)]^2}{\sum_{x_i^t \in \Psi_t}[f(x_i^t) - f_M(\Psi_t)]^2} \tag{5.17}$$

其中

$$f_M(\Psi_t) = \min \{f(x_i^t);\ x_i^t \in \Psi_t\} \tag{5.18}$$

3. 多点最速下降算法

他们把多初始点的最速下降法嵌入到遗传算法中，以提高传统遗传算法的性能，随机初始解的多点最速下降法可以描述为

多点最速下降法

步骤 1：初始化

随机产生初始解 x。

步骤 2：邻域搜索

随机地顺序检查当前解 x 邻域中的解，令 y^* 是第一个改善当前解的解。如果邻域中无更好的解，令 $y^* = \varnothing$，其中$\varnothing$表示 y^* 是空集。

步骤 3：终止条件检查

如果满足预先给定的终止条件，停止算法；否则，如果 y^* 为空，转步骤 1；如果 y^* 非空，令 $x := y^*$ 返回步骤 2；评估解的总数作为停止准则。

4. 混合式遗传算法

Ishibuchi-Yamamoto-Murata-Tanaka 方法的总体步骤可以总结如下：

Ishibuchi-Yamamoto-Murata-Tanaka 方法步骤

步骤 1：初始化。

步骤 2：下降搜索

取当前种群中的染色体作为初始解，应用多点最速下降法，再用得到的解更新当前种群。

步骤 3：选择。

步骤 4：基于位置的交叉。

步骤 5：移动变异。

步骤 6：选择策略

从当前种群中随机移出一个解，加入前一种群中的最好解。

步骤 7：终止条件检查

如果满足预先确定的终止条件，停止算法；否则转步骤 2。

5.6.3 计算实例

Ishibuchi 等人就规模为 10～50 个工件的随机产生的问题对混合式遗传算法和其他方法进行了比较[233]，比较结果如表 5.9 所示。结果说明遗传算法优于其他方法。

表 5.9 遗传算法与其他方法结果的比较

问题规模 (工件数)	检查染色体 总数	GA	DA	TS	SA	GDA
10 个工件	10 000	0.49	0.50	0.50	0.50	0.50
	100 000	0.50	0.50	0.50	0.50	0.50
20 个工件	10 000	0.34	0.40	0.39	0.39	0.41
	100 000	0.40	0.47	0.49	0.47	0.49
50 个工件	10 000	0.00	0.04	0.01	0.10	0.04
	100 000	0.11	0.18	0.15	0.26	0.24

注：GA：遗传算法；DA：多点最速下降算法；
TS:禁忌搜索；SA：模拟退火；
GDA：嵌入 DA 的遗传算法。

第六章 作业车间调度问题

6.1 引 言

机器调度问题来源于不同的领域，如柔性制造系统，生产计划，计算机设计，后勤及通信等，这些问题的共同特性是没有一个有效的算法能在多项式时间内求出其最优解。古典的作业车间调度问题是最著名的机器调度问题之一。问题可以描述为：给定一个工件的集合和一个机器的集合，每个工件包括多道工序，每道工序需要在一台给定的机器上非间断地加工某一段时间；每台机器一次最多只能加工一道工序；调度就是把工序分配给机器上某个时间段．问题的目标是找到最小时间长度的调度。在过去的 30 年里，机器调度问题吸引了无数研究者的浓厚兴趣，大量的研究成果相继问世，其中包括 Muth 和 Thompson[307]，Conway 等[84]，Baker[21]，French[44]，Rinnooy Kan[356]，Coffman[78]，Blazewicz 等[41]的著作，以及 Morton 和 Pentico 的最新著作[299]。

作业车间调度问题(JSP)是最困难的组合最优化问题之一[148]。由于其本身的难处理的特性，一些启发式算法成为有吸引力的备选方法。多数传统的启发式算法用优先权规则。所谓优先权规则是一个从未排序的工序特定子集中选用工序的规则。近年来，概率的局域搜索方法的发展引起了人们用局域搜索方法求解作业车间调度问题的兴趣，如模拟退火(SA)[409]，禁忌搜索(TS)[107]，遗传算法(GA)[15,86,100,112,122,124,160,165,221,313,332,363,378,397,423,460]。近年来用遗传算法求解古典作业车间调度问题的文章见表 6.1。

表 6.1 有关 GA/JSP 的最近文章

作　者	描述方法
Davis(1985)[100]	基于优先列表表达法
Nakano 等(1991)[313]	基于工件对关系表达法
Falkenauer 等(1991)[122]	基于优先列表表达法
Bagchi 等(1991)[15]	特定问题表达法
Yamada 等(1992)[423]	基于完成时间表达法
Tamaki 等(1992)[397]	基于非连接图表达法
Paredis(1992)[332]	基于工件对关系表达法
Holsapple 等(1993)[221]	基于工件表达法
Fang 等(1993)[124]	基于工序表达法
Gen 等(1994)[160]	基于工序表达法
Bean(1994) [29]	随机键表达法
Dorndorf 等(1995)[112]	基于优先规则和基于机器的表达法
Croce 等(1995)[86]	基于优先列表表达法
Kobayashi 等(1995)[249]	基于优先列表表达法
Norman 等(1995)[317]	随机键表达法

在本节中，我们回顾用遗传算法求解古典作业车间调度问题的最新文献，详细解释一些典型遗传算法的实现方法。

6.2 古典作业车间模型

古典作业车间调度问题可以表述为：有 m 台不同的机器和 n 个不同 的工件，每个工件包含一个由多道工序组成的工序集合，工件的工序顺序是预先给定的。每道工序用它所要求的机器和固定的加工时间来表示。此外对工件和机器有一些约束，例如

(1) 一个工件不能两次访问同一机器；

(2) 不同工件的工序之间没有先后约束；

(3) 工序一旦进行不能中断；

(4) 每台机器一次只能加工一个工件；

(5) 下达时间和交货期都不是给定的。

问题是确定机器上工序的顺序使最大流程时间，即完成所有工件的时间，达到最小。

让我们看一个由表 6.2 给出的 3 个工件、3 台机器的问题。我们用甘特图求解其可行的调度，有两类甘特图：机器甘特图和工件甘特图。用下标 jom 来命名每一道工序，这里 j 指出工件，o 表示工件 j 的工序，m 表示加工这道工序的机器。例如，下标 321 表示工件 3 的第 2 道工序在机器 1 上进行。图 6.1 表示每台机器上不同工件可能的加工时间，而图 6.2 表示了每个工件的多道工序可能进行的时间。

表 6.2 3 个工件、3 台机器的问题

加工时间				机器序列			
工件	工序			工件	工序		
	1	2	3		1	2	3
j_1	3	3	2	j_1	m_1	m_2	m_3
j_2	1	5	3	j_2	m_1	m_3	m_2
j_3	3	2	3	j_3	m_2	m_1	m_3

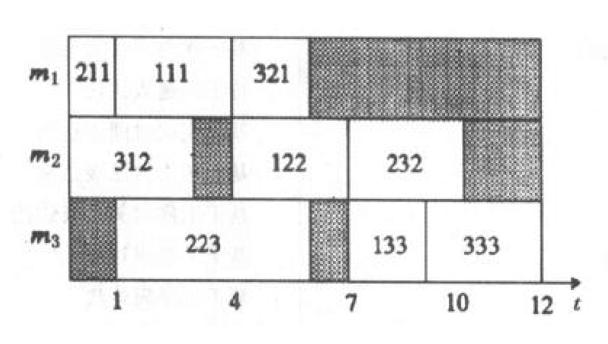


图 6.1 机器甘特图

原则上，作业车间问题有无数个可行的调度，因为两道工序之间可以插入不必要的闲置时间。我们可以尽可能压缩工序往左移动。如果一些工序提前开工而不改变工序的顺序，那么这个调度中的移动称为局部左移动(local left-shift)。如果一些工序提前开工虽然改变了工序的顺序，但没有延迟任何其他的工序，那么这个调度中的移动称为全局左移

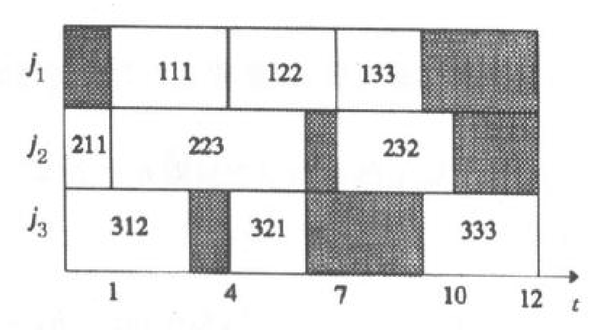


图 6.2　工件甘特图

动(global left-shift)。基于以上两个概念，有以下三类不同的调度：

定义 6.1(半活动调度)　一个调度是半活动的，如果不存在局部左移动。

定义 6.2(活动调度)　一个调度是活动的，如果不存在全局左移动。

定义 6.3(无延迟调度)　一个调度是无延迟的，如果没有机器在能够开始加工时处于闲置状态。

Venn 给出的活动调度，半活动调度和无延迟调度的关系如图 6.3 所示。最优调度在活动调度集中。无延迟调度集小于活动调度集，但不能保证包括最优解[21]。

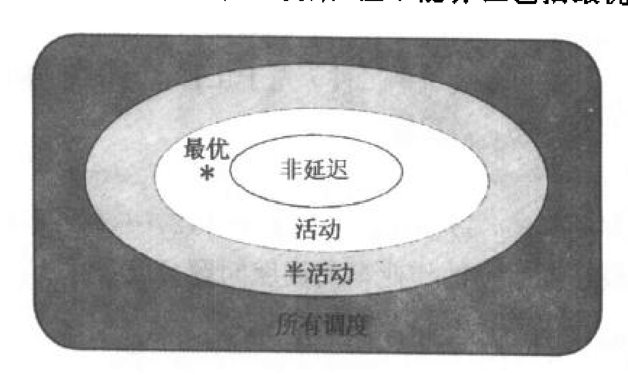


图 6.3　Venn 的调度关系图

6.2.1　整数规划模型

Baker 在文献[21]中讨论了单件车间调度问题的整数规划(IP)模型，Greenberg 在文献[184]中更完整地讨论了 IP 描述。IP 描述依赖于指示变量来指定工序的顺序；Cheng，Gen 和 Tsujimura 提出了一种新的整数规划描述，它用一个线性不等式系统给出了一个表达优先约束的更好方法[71]。

古典作业车间调度问题需要考虑两类约束：

(1) 给定工件的工序前后约束；

(2) 给定机器的工序非堵塞约束。

一般地，令 c_{jk} 表示工件 j 在机器 k 上的完工时间，t_{jk} 表示工件 j 在机器 k 上的加工时间。

首先考虑给定作业车间调度的工序前后约束。

对于工件 i，如果在机器 h 上的加工先于机器 k，有如下约束：

$$c_{ik} - t_{ik} \geqslant c_{ih}$$

另一方面，如果在机器 k 上的加工首先出现，有约束

$$c_{ik} - t_{ik} \geqslant c_{ik}$$

由于约束中的一个或多个都必须满足，因此称它们为分离性约束(Disjunctive Constraints)。定义如下的指示系数：

$$a_{ihk} = \begin{cases} 1, & \text{对于给定的工件 } i\text{，如果在机器 } h \text{ 上的加工先于机器 } k \\ 0, & \text{其他} \end{cases}$$

以上约束也可以写成

$$c_{ik} - t_{ik} + M\,(1 - a_{ihk}) \geqslant c_{ih}; \quad i = 1,2,\cdots,n; \quad h,\, k = 1,2,\cdots,m$$

其中 M 是一个很大的数。注意上面的不等式很巧妙地吸收了前后约束。如果先在机器 h 上加工，$M\,(1 - a_{ihk})$为 0，给定的不等式正是所要的。否则，$M(1 - a_{ihk})$变得很大，不等式同样为真。

现在来考虑给定机器的工序非堵塞约束：

对于两个工件 i 和 j 都需要在机器 k 上加工，如果 i 先于工件 j 来到，有以下约束：

$$c_{jk} - c_{ik} \geqslant t_{jk}$$

另一方面，如果工件 j 先来到，有约束：

$$c_{ik} - c_{jk} \geqslant t_{ik}$$

定义指示器变量 x_{ijk}为

$$x_{ijk} = \begin{cases} 1, & \text{如果工件 } i \text{ 先于工件 } j \text{ 来到机器 } k \\ 0, & \text{其他} \end{cases}$$

以上约束可以写成

$$c_{jk} - c_{ik} + M\,(1 - x_{ijk}) \geqslant t_{jk}; \quad i,\, j = 1,2,\cdots,n; \quad k = 1,2,\cdots,m$$

以最大流程时间最小为目标的作业车间调度问题可以表述为

$$\min \quad \max_{1\leqslant k\leqslant m}\left\{ \max_{1\leqslant i\leqslant n}\{c_{ik}\}\right\} \tag{6.1}$$

$$\text{s.t.} \quad c_{ik} - t_{ik} + M\,(1 - a_{ihk}) \geqslant c_{ih}; \quad i = 1,2,\cdots,n; \quad h,\, k = 1,2,\cdots,m \tag{6.2}$$

$$c_{jk} - c_{ik} + M\,(1 - x_{ijk}) \geqslant t_{jk}; \quad i,\, j = 1,2,\cdots,n; \quad k = 1,2,\cdots,m \tag{6.3}$$

$$c_{ik} \geqslant 0, \quad i = 1,2,\cdots,n; \quad k = 1,2,\cdots,m \tag{6.4}$$

$$x_{ijk} = 0 \text{ 或 } 1; \quad i,\, j = 1,2,\cdots,n; \quad k = 1,2,\cdots,m \tag{6.5}$$

约束(6.2)保证每个工件的工序的加工顺序满足预先的要求；约束(6.3)保证每台机器一次只能加工一个工件。

如果考虑平均流程时间问题，目标可以写成：

$$\min \quad \sum_{i=1}^{n} \max_{1\leqslant k\leqslant m}\{c_{ik}\} \tag{6.6}$$

6.2.2 线性规划模型

Adams 等人在文献[3]中讨论了作业车间调度问题的线性规划(LP)模型。令 $N=\{0,1,2,\cdots,n\}$表示工序的集合，其中 0 和 n 表示虚设的起始工序和完成工序；$M=\{1,\,2,\,\cdots,\,m\}$是机器的集合；A 是表示每个工件的工序前后关系约束的工序对集合；E_k 是机器 k 上加工的工序对集合。对于每个工序 i，加工时间 d_i 是一定的，工序的起始时间 t_i 是优化过程中有待确定的变量。因此，作业车间调度问题可以描述为

$$\min \quad t_n \tag{6.7}$$

$$\text{s.t.} \quad t_j - t_i \geqslant d_i, \quad (i, j) \in A \tag{6.8}$$

$$t_j - t_i \geqslant d_i \text{ 或 } t_i - t_j \geqslant d_j, \quad (i, j) \in E_k, \quad k \in M \tag{6.9}$$

$$t_i \geqslant 0 \tag{6.10}$$

约束(6.8)保证每个工件的工序顺序满足预先的要求，约束(6.9)保证每台机器一次只能加工一个工件。问题的一个可行解称为一个调度。

6.2.3 图模型

作业车间调度问题可以用非连接图来表示[23,360]，非连接图 $G=(N,A,E)$ 定义为：N 包含代表所有工序的节点，A 包含连接同一工件的邻接工序的边，E 包含连接同一机器上加工工序的非连接边，所谓非连接边是可以有两个可能方向的边。调度将固定所有非连接边的方向，以确定同一机器上的工序顺序；一旦对于某台机器的顺序确定下来，连接工序的非连接边将被通常的带优先箭头的连接边取代。非连接边集 E 分解成子集系 E_k；$E=E_1\cup E_2\cup E_3\cdots\cup E_m$，每台机器一个系。图 6.4 是 3 个工件、3 台机器的非连接图实例，其中每个工件包含 3 道工序，节点 $N=\{0,1,2,3,4,5,6,7,8,9,10\}$ 对应工序，其中 0 和 10 是虚设的起始工序和终止工序。连接边 $A=\{(1,2),(2,3),(4,5),(5,6),(7,8),(8,9)\}$ 对应同一工件的工序前后约束。非连接边(虚线表示)$E_1=\{(1,5),(5,9),(9,1)\}$ 对应机器 1 上加工的工序，非连接边 $E_2=\{(4,2),(2,8),(8,4)\}$ 对应机器 2 上加工的工序，非连接边 $E_3=\{(7,3),(3,6),(6,7)\}$ 对应机器 3 上加工的工序。

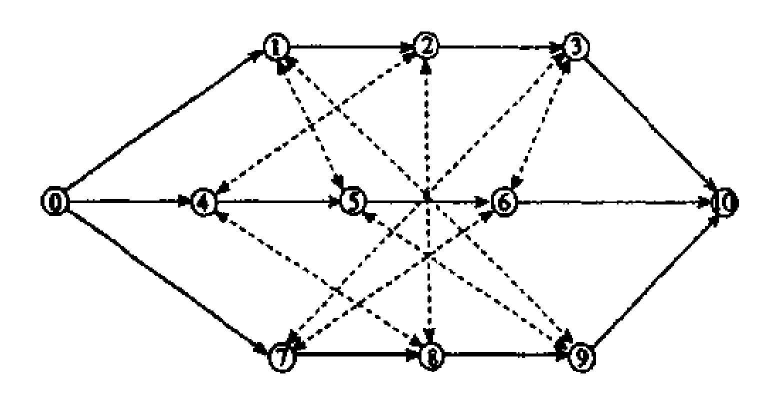


图 6.4 非连接图表达法

作业车间调度问题是找出每台机器上工序的加工顺序，即确定非连接边的方向，使得产生的结果图是非循环的(工序间无先后冲突)，并且起始点到终止点的最大权路径的长度最小。最大权路径的长度对应于最大流程时间。

6.3 传统启发式方法

作业车间调度问题是很重要的常见实际问题。由于作业车间调度问题是最困难的组合优化问题，因此很自然地希望寻找一些近似方法能在有效时间内产生可接受的调度。作业车间调度问题的启发式方法大致分为两类：

(1) 一次通过启发式；

(2) 多次通过启发式。

一次通过启发式指通过基于优先分配规则一次固定一个工序，简单地构造一个完全解。有多种规则可用来从特定的子集中选择工序作为下一步的调度。这种启发式的特点是快，而且能找到不太坏的解。此外，可以重复应用一次通过启发式来建立更为复杂的多次通过启发式，以花费额外的计算费用来获得更好的调度。

6.3.1 优先分配启发式方法

由于优先分配启发式具有容易实现和较小的时间复杂性，因而是实际求解调度问题经常使用的方法，Giffler 和 Thompson 算法被视为所有基于优先规则启发式的共同基础[389]。Giffler 和 Thompson 提出了构造调度的两个算法：活动调度法和无延迟调度法[167]。无延迟调度具有如下性质：如果有工件等待加工，则没有机器处于闲置状态。活动调度具有如下性质：没有工序在不延迟其它工件的情况下而能够提前开工。活动调度构成一个包括无延迟调度作为子集的更大的集合。Giffler 和 Thompson 构成调度的方法是树形结构方法，树中的节点对应部分调度，边对应可能的选择，树的叶子是可数的调度的集合。对于一个给定的部分调度，算法主要是识别所有的加工冲突，即竞争同一机器的工序，尔后在每一阶段采取一些步骤以多种可能的方式解决这些冲突。而启发式则用优先分配规则，即在冲突的工序中按优先规则选择工序来解决这些冲突。

在每一阶段用可调度的工序的集合构造调度。可调度的工序是指迄今尚未调度而其紧前工序刚调度过的工序，可以从工序的衔接结构中确定。一次通过法的阶段数等于工序数 $m\times n$。在每一阶段，选择一个工序加入到部分调度中，工序之间的冲突用优先分配规则解决。根据 Baker 的有关概念[21]，有

PS_t——包含 t 个已调度工序的部分调度；

S_t——阶段 t 可调度工序的集合，对应给定的 PS_t；

σ_i——工序 $i\in S_t$ 能够开始的最早时间；

ϕ_i——工序 $i\in S_t$ 能够完成的最早时间。

对于给定的活动部分调度，可能的起始时间 σ_i 由工序 i 的直接紧前工序的完成时间和工序 i 要求的机器的最迟完成时间确定。设两个值中较大的一个是 σ_i，可能完成的时间为 $\phi_i=\sigma_i+t_i$，其中 t_i 是工序 i 的加工时间。构造活动调度的步骤如下：

优先分配启发式算法（产生活动调度）

步骤 1：令 $t=0$，开始时 PS_t 为空部分调度。初始的 S_t 包括无紧前工序的所有工序。

步骤 2：确定 $\phi_t^*=\min_{i\in S_t}\{\phi_i\}$ 和 能实现 ϕ_t^* 的机器 m^*。

步骤 3：对于每一道要求使用机器 m^* 的工序 $i\in S_t$ 且 $\sigma_i<\phi_t^*$，根据预先给定的优先规则计算优先指标，把具有最小指标的工序尽可能早地加入到 PS_t，构造一个新的部分调度 PS_{t+1}。

步骤 4：对于 PS_{t+1}，按如下步骤更新数据集：

1) 从 S_t 中删除工序 i；

2) 通过在 S_t 中增加工序 i 的直接后继工序构造 S_{t+1}；

3) $t=t+1$。

步骤 5：返回步骤 2 直至构造一个完整的调度。

如果我们在以上算法中用以下算法代替步骤 2 和步骤 3，那么我们就能得到无延迟调度。

优先分配启发式算法(产生无延迟调度)

步骤 1：令 $t=0$，开始时 PS_t 是空部分调度。初始的 S_t 包括无紧前工序的所有工序。

步骤 2：确定 $\sigma_t^* = \min_{i\in S_t}\{\sigma_i\}$ 和 σ_t^* 能实现 σ_t^* 的机器 m^*。

步骤 3：对于每一道要求使用机器 m^* 的工序 $i\in S_t$ 且 $\sigma_i = \sigma_t^*$，根据预先给定的优先规则，计算优先指标。把具有最小指标的工序尽可能早地加入到 PS_t，构造一个新的部分调度 PS_{t+1}。

步骤 4：对于 PS_{t+1}，更新数据集如下：

1) 从 S_t 中删除工序 i；

2) 通过在 S_t 中增加工序 i 的直接后继工序构造 S_{t+1}；

3) $t=t+1$。

步骤 5：返回步骤 2 直至构造一个完整的调度。

剩下的问题是识别一个有效的优先规则。进一步的结论和讨论可参见 Panwalkar 和 Iskander[331]，Haupt[210]，Blackstone 等人[38]的工作。表 6.3 包含了实际中常用的一些优先规则。

表 6.3 作业车间分配规则

规则	描述
SPT	优先选择最短加工时间的工序
LPT	优先选择最长加工时间的工序
MWR	优先选择剩余总加工时间最长的工件的工序
LWR	优先选择剩余总加工时间最小的工件的工序
MOR	优先选择剩余工序数最多的工件的工序
LOR	优先选择剩余工序数最小的工件的工序
EDD	优先选择具有最早交货期的工件
FCFS	选择同一机器上工件队列中的第一道工序
RANDOM	随机选择工序

6.3.2 随机分配启发式

一次通过启发式局限于构造一个解，多次通过启发式(也称为搜索启发式)试图通过产生多个解来得到更好的解，却常常以损失计算时间为代价。而分枝定界法和动态规划法尽管能够保证最优解，但不能实用于大规模的系统。

随机化的方法是提供更精确解的早期方法[21]。随机分配的思想是从一族分配规则出发，在选择每一道工序运行时，随机选择分配规则，重复进行直到产生全部调度。多次重复整个过程，并选择最好解。产生活动调度的过程如下：

随机分配启发式算法(产生活动调度)

步骤 1：令最好的调度 BS 为空调度。

步骤 2：令 $t=0$ 以 PS_t 作为空的部分调度。初始 S_t 包括无紧前工序的所有工序。

步骤 3：确定 $\phi_t^*=\min_{i\in S_t}\{\phi_i\}$ 和能够实现 ϕ_t^* 的机器 m^*。

步骤 4：从一族规则中随机选择一个分配规则，对于每一道要求使用机器 m^* 的工序 $i\in S_t$ 且 $\sigma_i<\phi_t^*$，根据预先选定的规则，计算优先指标。把具有最小指标的工序尽可能早地放入 PS_t，产生新的部分调度 PS_{t+1}。

步骤 5：对于 PS_{t+1}，更新数据集：

1）从 S_t 中删除工序 i；

2）把工序 i 的直接后继工序加入到 S_t，构成 S_{t+1}；

3）$t=t+1$。

步骤 6：返回到步骤 3 直至产生一个完整的调度。

步骤 7：如果当前产生的解优于 BS，更新 BS。返回到步骤 2，直至预先设置的迭代步数。

许多研究者曾对随机化的方法作过改进，其中之一是加入学习过程，使得更好的分配规则有更多的机会被选择。Morton 和 Pentico 提出了一个引导随机方法[299]。这里的“引导”意味着首先需要一个很好的启发式来发掘问题并能引导搜索。

6.3.3 瓶颈移动启发式方法

瓶颈移动启发式(shifting bottleneck heuristic)源于 Adams，Balas 和 Zawack 的文献[3]，是目前作业车间调度问题最有效的启发式方法。机器一个接一个地逐次调度，每次调度时把机器视为未调度机器当中的瓶颈，每新调度一部机器后、所有以前的调度都要进行重新局部优化。瓶颈的识别和重新局部优化过程都是基于求解一个单机的调度问题，它是原问题的松弛问题。尽管 Adams，Balas 和 Zawack 显著地加速了产生这些问题需要的时间，但没有提出求解单机问题的新方法。该方法的主要贡献在于用松驰方式来决定机器应该排列的顺序。其思想仍是基于赋瓶颈机器优先权(数)的思想。令 M_0 是已排好序机器的集合(开始时，$M_0=\varnothing$)，瓶颈移动方法的简单过程如下：

瓶颈移动启发式算法

步骤 1：在机器 $k\in M\setminus M_0$ 中识别瓶颈机器 m，最优化调度机器 m。置 $M_0\leftarrow M_0\cup\{m\}$。

步骤 2：保持其它机器的顺序不变，重新依次优化每个关键机器 $k\in M_0$，如果 $M_0=M$，停止；否则，转步骤 1。

算法实现的详细过程可参见文献[3]或文献[9]。根据 Adams，Balas 和 ZawacK 的计算经验，一般的改进量在 4%～10%之间。Storer，Wu 和 Vaccari 曾指出 Applegate 和 Cook 实现瓶颈移动启发式的方法不能求解两个 10×50 问题[389]。近来，Dauzere-Peres 和 Lasserre 给出了一个改进的瓶颈移动启发式方法[92]。

6.4 作业车间调度问题的遗传算法

6.4.1 表达方法

由于受工序的加工路线的约束，作业车间调度问题不像旅行商问题(TSP)那样容易

确定一个自然表达。到目前为止，还没有一个用系统不等式来表达先后约束的好方法。因此，不便于应用惩罚法来处理这些类型的约束。Orvosh 和 Davis 的研究[322]表明，对于很多组合优化问题，修复非可行或非法的染色体相对容易些，修复的策略确实要优于其它诸如删除策略和惩罚策略。多数的遗传算法和作业车间调度问题的研究者喜欢使用修复策略来处理非可行性和非法性。构造作业车间问题遗传算法的一个非常重要的主题是设计一个合适的表达解的方法和基于特定问题的遗传算子，使得不管在初期还是在进化过程中产生的所有染色体都将产生可行的调度，而这将是影响遗传算法的各个子阶段的关键阶段。在过去的几年里，已经提出了以下 9 种作业车间的表达方法。

(1) 基于工序的表达法；

(2) 基于工件的表达法；

(3) 基于优先表的表达法；

(4) 基于工件对关系的表达法；

(5) 基于优先规则的表达法；

(6) 基于非连接图的表达法；

(7) 基于完成时间的表达法；

(8) 基于机器的表达法；

(9) 随机键表达法。

以上这些表达法可以归为如下两类基本的编码方法：

(1) 直接方法；

(2) 间接方法。

在直接法中，一个调度被编码为一个染色体，应用遗传算法来进化染色体以确定一个好的调度。基于工序表达法，基于工件表达法，基于工件对关系表达法，基于完成时间法和随机键表达法都属于此类。

在间接法中，例如基于优先规则表达法，工件调度的分配规则序列被编码为一个染色体，应用遗传算法来进化染色体以确定一个好的分配规则序列。基于优先表的表达法，基于优先规则表达法，基于非连接图表达法和基于机器表达法都属于此类。

我们将用表 6.2 给出的简单实例依次讨论这些方法。

1. 基于工序的表达法

这种表达法把调度编码为工序的序列，每个基因代表一道工序。它有两种可能的方式来命名每道工序，一种自然的方式是用自然数来命名工序，像 TSP 的顺序表达法，遗憾的是由于受加工路线的约束，不是所有的自然数的排列都可以定义可行的调度。Gen，Tsujimura 和 Kubota 提出了另一种方式：给所有同一工件的工序指定相同的符号，然后根据它们在给定染色体中出现的顺序加以解释[105,256]。对于 n 工件 m 机器问题，一个染色体包括 $n\times m$ 个基因。每个工件出现在染色体中 m 次，每个基因不表明一个工件的具体的工序，而是指有上下依赖关系的工序。很容易看出染色体的任意排列总能产生可行调度。

考虑表 6.2 所示的 3 个工件、3 台机器问题。假设给定染色体为[3 2 2 1 1 2 3 1 3]，其中 1 表达工件 j_1，2 代表工件 j_2，3 代表工件 j_3。因为每个工件有三道工序，所以每个

工件在一个染色体中刚好出现三次。例如，在以上给定的染色体中出现三个 2 表示工件 j_2 的三道工序，第一个 2 对应工件 j_2 第一道工序在机器 1 上加工，第二个 2 对应工件 j_2 的第二道工序在机器 3 上加工，第三个 2 对应工件 j_2 的第三道工序在机器 2 上加工。我们可以看到工件 j_2 的所有工序都用相同的符号 2 来命名，并且根据它出现染色体中的顺序得到解释，工件的工序和加工机器的对应关系如图 6.5 所示。根据对应关系，我们可以得到相应的机器列表为[2 1 3 1 2 2 1 3 3]，如图 6.6 所示。从图中可以看到，机器 1 上工件的加工顺序为 2 - 1 - 3(阴影部分)，机器 2 上工件的加工顺序为 3 - 1 - 2，机器 3 上工件的加工顺序为 2 - 1- 3，由此可以得到一个可行调度，如图 6.7 所示。

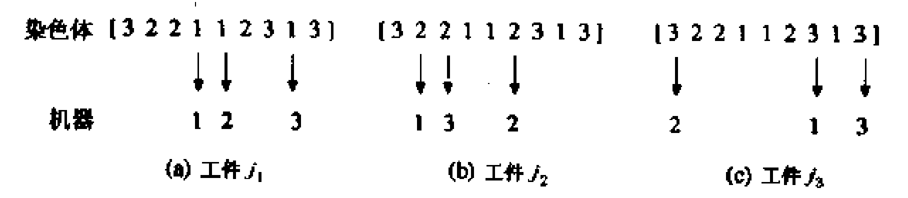


图 6.5 工件的工序与相应的机器

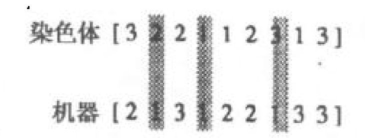


图 6.6 机器 1 上工件的加工顺序

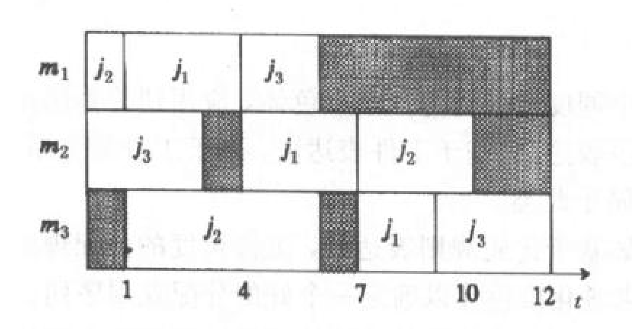


图 6.7 一个可行调度

Fang，Ross 和 Corne 也提出了一种作业车间调度问题的基于工序表达法[124]。对于一个 $j\times m$ 问题，染色体是包含 $j\times m$ 个块(chunk)的串(string)，足够容纳最大的工件数 j。一个块即相当于遗传算法的原子(atomic)，无疑一块对应一道工序，但该文中给出的关于“块”的描述不够清晰，以至我们不能把一道工序编码成一块。当我们构造一个实际的调度时，这些方法保持一个非完工工件的环形列表和多个对于这些工件的未调度的工序列表。一块提供了关于在这些列表中如何找到下一个要调度工序的信息。块可用作非完工工件的环形列表中工件的下标，这样，工件的第一道未处理的工序将被排列到尽可能早的位置。

2. 基于工件的表达法

这种表达法由包含 n 个工件的列表组成，每个调度根据工件的顺序来构造。对于一个给定的工件顺序，列表中第一个工件的所有工序将被首先调度，然后考虑列表中第二

个工件的工序。加工中的工件第一道工序要求的加工时间应为相应机器最可能得到的加工时间，然后考虑第二道工序，如此进行，直到工件的所有工序排完。这个过程以列表中每个工件的适当顺序重复进行。工件的任意序列对应一个可行的调度。Holsapple，Jacob，Pakath 和 Zaveri 已经用这种表达法处理过柔性制造系统的静态调度问题[221]。

我们同样用表 6.2 中给出的 3 个工件、3 台机器的问题来说明。假设给定染色体为[2 3 1]。要加工的第一个工件是 j_2。工件 j_2 的工序前后约束是[$m_1\ m_3\ m_2$]，每台机器的相应的加工时间为[1 5 3]。首先工件 j_2 按图 6.8 (a)调度，然后加工工件 j_3，它在机器中的工序前后约束为[$m_2\ m_1\ m_3$]，每台机器的 相应加工时间为[3 2 3]，其每道工序安排在相应机器最可能得到的加工时间进行，如图 6.8(b)所示。最后工件 j_1 按图 6.8(c)所示进行加工。

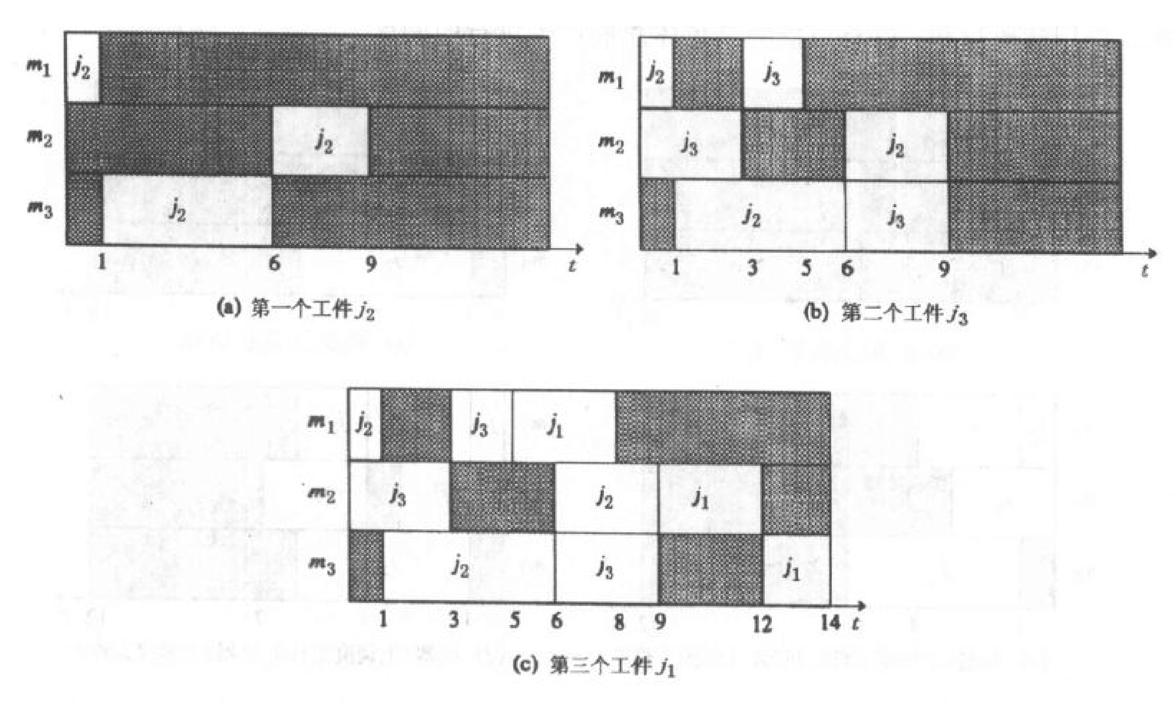


图 6.8 调度工件

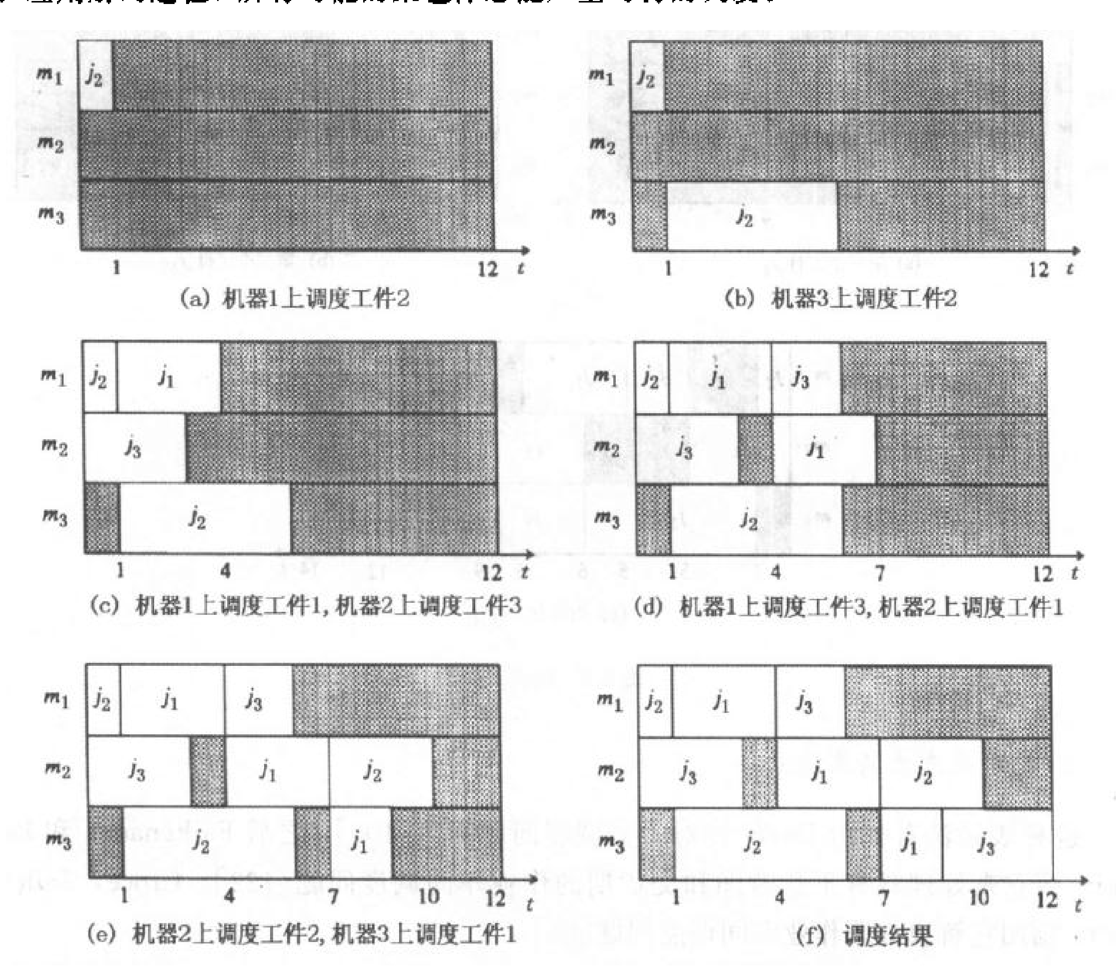
3. 基于优先表的表达法

这种表达法首先由 Davis 针对一类调度问题提出[100]，之后 Falkenauer 和 Bouffouix 用它来处理具有下达时间和交货期的作业车间调度问题[122]。Croce，Tadei 和 Volta 则用它解决古典作业车间调度问题[86]。

对于 n 个工件 m 台机器的作业车间调度问题，一个染色体由 m 个子染色体组成，每个子染色体对应一台机器，由一串长度为 n 的符号表示，每个符号代表一道在相关机器上处理的工序。子染色体不能描述机器上工序的先后顺序，因为它们是优先列表；每台机器有其自身的优先列表。实际的调度是由染色体通过模拟分析机器前等待排队的状态得到的，如果有必要用优先列表来确定调度，即选择首先出现在优先列表中的工序。

下面我们通过一个实例来说明如何由给定的染色体导出实际的调度。考虑由表 6.2 给出的相同实例。给定染色体为 [(2 3 1) (1 3 2) (2 1 3)]。第一个基因(2 3 1)是机器 m_1 的优先列表，(1 3 2)是机器 m_2 的优先列表，(2 1 3)是机器 m_3 的优先列表。从

这些优先列表可知第一道优先的工序是机器 m_1 上加工工件 j_2，机器 m_2 上加工工件 j_1，机器 m_3 上加工工件 j_2。根据给定的工序先后约束，只有机器 m_1 上加工工件 j_2 是可行的调度，因此首先对机器 m_1 调度，如图 6.9(a)所示。第二道可调度的工序是在机器 m_3 上加工工件 j_2，如图 6.9(b)所示。当前的优先工序是在机器 m_1 上加工工件 j_3，机器 m_2 和 m_3 上加工工件 j_1。由于这些工序在当前时间都是非可行的，所以只好从列表中找第二级优先工序，即机器 m_1 上加工工件 j_1，机器 m_2 和 m_3 上加工工件 j_3。可调度的工序是机器 m_1 上加工工件 j_1，机器 m_2 上加工工件 j_3，如图 6.9(c)所示。下一道可调度的工序是机器 m_1 上加工工件 j_3，机器 m_2 上加工工件 j_1，如图 6.9(d)所示。然后可调度的工序是机器 m_2 上加工工件 j_2，机器 m_3 上加工工件 j_1，如图 6.9(e)所示。最后一道工序是机器 m_3 上加工工件 j_3，如图 6.9(f)所示。以上就是从染色体中得到一个流程时间为 12 的可行调度的过程。应用解码过程，所有可能的染色体总能产生可行的调度。



(a) 机器1上调度工件2　(b) 机器3上调度工件2

(c) 机器1上调度工件1, 机器2上调度工件3　(d) 机器1上调度工件3, 机器2上调度工件1

(e) 机器2上调度工件2, 机器3上调度工件1　(f) 调度结果

图 6.9　从染色体中得到一个调度的过程

Croce 等人曾经认为推导过程只是产生非延迟的调度，因此不能确保其中包含有最优解[21]。他们给出了一个更为复杂的超前(lookahead)评估过程来帮助推导过程，以得到一个活动的调度。可惜这种观点是不正确的。当产生一个非延迟的调度时，我们必须首先识别一个能够最早开工的机器(关键机器)，并选择一道能够在关键机器上最早加工的工序。因为在推导过程中，要根据优先列表选择下一道工序，所以可能得不到一个非延迟的调度。推导过程能确保产生一个活动调度，例如，图 6.9(f)给出的调度是一个活动

调度，不是非延迟调度。因为机器 m_3 在工件 j_3 的第三道工序开始时刻 6 处于闲置状态。

Kobayashi，Ono 和 Yamamura 在他们的研究[249]中也采用过这种表达法。不同之处是他们用 Giffler 和 Thompson 的启发式算法解码染色体，来得到一个调度。

4. 基于工件对关系的表达法

Nakano 和 Yamada 采用二进制矩阵编码一个调度。矩阵是根据相应机器上工件对的前后关系来确定的[313]。

考虑 3 个工件、3 台机器的调度问题。工件的工序前后约束和问题的一个可行调度如表 6.4 所示。

表 6.4 3个工件、3台机器的实例

工序先后约束				可行调度			
工件	机器顺序			机器	工件调度		
j_1	m_1	m_2	m_3	m_1	j_2	j_1	j_3
j_2	m_1	m_3	m_2	m_2	j_3	j_1	j_2
j_3	m_2	m_1	m_3	m_3	j_2	j_1	j_3

用一个二进制变量来标识工件对的先后关系，定义如下：

$$x_{ijm} = \begin{cases} 1, & \text{如果工件 } i \text{ 在机器 } m \text{ 上加工优先于工件 } j \\ 0, & \text{其他} \end{cases}$$

我们首先检验工件对(j_1，j_2)在机器 (m_1，m_2，m_3)的先后关系。根据给定的调度，有 (x_{121} x_{122} x_{123}) = (0 1 0)。对于工件对(j_1，j_3)，有(x_{131} x_{132} x_{133}) = (1 0 1)；对于工件对(j_2,j_3)，在机器 (m_1，m_3，m_2)的先后关系为(x_{231} x_{233} x_{232}) = (1 1 0)。值得注意的是，对于一个工件对，变量 x_{ijm}的排列顺序应该跟第一个工件 i 的工序顺序一致。例如对于工件对(j_2，j_3)，工件 j_2 的工序顺序为(1 3 2)，因此相对变量排列为(x_{231} x_{233} x_{232})，而不是(x_{231} x_{232} x_{233})。总结以上结果，对于给定的可行调度可以得到一个二进制矩阵表达法：

$$\begin{matrix} (j_1, j_2) \text{ 关于 } (m_1,m_2,m_3): \\ (j_1, j_3) \text{ 关于 } (m_1,m_2,m_3): \\ (j_2, j_3) \text{ 关于 } (m_1,m_3,m_2): \end{matrix} \begin{bmatrix} x_{121} & x_{122} & x_{123} \\ x_{131} & x_{132} & x_{133} \\ x_{231} & x_{233} & x_{232} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

二进制矩阵表达法也许是这些表达法中最复杂的一种，而且有高的冗余[124]；另一个问题是无论初始步骤还是遗传运算产生的染色体一般都是非法的。

5. 基于优先规则的表达法

Dorndorf 和 Pesch 在文献[112]中提出了一种基于优先规则的遗传算法，其中染色体编码为一个工件分配规则的序列。基于分配规则序列，用优先分配启发式构造调度，遗传算法用于进化染色体以方便产生一个好的分配规则序列。

由于该算法便于实现，且时间复杂性较小，优先分配规则可能是实际求解调度问题最多的启发式方法。Giffler 和 Thompson 的算法被认为是所有基于优先分配规则启发式

的基础[167,389]。主要问题是要寻找一个有效的分配规则。关于优先分配规则进一步的结果和讨论参见文献[331]，[210]和[38]。

对于 n 个工件、m 台机器的调度问题，染色体是一个 $n\times m$ 项的串(p_1, p_2, …, p_{nm})。p_i 代表预先设定的优先分配规则集合的一个规则。处于第 i 个位置的元素表达在 Giffler 和 Thompson 算法的第 i 步迭代中的冲突应该由优先规则 p_i 来重新处理。更精确地说，用规则 p_i 从冲突集合中选取工序，采用随机选择来断开链。令

PS_t——包含 t 步已调度工序的部分调度；

S_t——对应于 PS_t 在迭代 t 时可调度工序的集合；

σ_i——工序 $i\in S_t$ 能够开始的最早时间；

ϕ_i——工序 $i\in S_t$ 能够完成的最早时间；

C_t——迭代 t 时冲突工序的集合。

从给定染色体(p_1, p_2, …, p_{nm})得到一个调度的步骤为

基于优先规则编码的调度过程

步骤 1：令 $t=1$，PS_t 为空，S_t 包含所有无紧前工序的工序。

步骤 2：确定 $\phi_t^*=\min_{i\in S_t}\{\phi_i\}$ 和能够实现 ϕ_t^* 的机器 m^*；如果存在多个这样的机器，则通过随机选择来断开链。

步骤 3：形成一个包含需求机器 m^*，且满足 $\sigma_i<\phi_t^*$ 的工序 $i\in S_t$ 的冲突集合 C_t；从 C_t 中根据优先规则 p_t 选择一道工序尽可能早地添加到 PS_t 中，产生一个新的部分调度 PS_{t+1}。如果根据优先规则，p_t 中存在多道可选工序，则通过随机选择来断开链。

步骤 4：从 S_t 中移出被选择的工序，将直接后继工序添加到 S_t，更新 PS_{t+1}。$t=t+1$。

步骤 5：继续步骤 2 直到产生一个完全调度。

下面用表 6.5 给出的四个优先规则来说明。

表 6.5　选择的优先规则

序号	规则	描述
1	SPT	选择具有最短加工时间的工序
2	LPT	选择具有最长加工时间的工序
3	MWR	选择剩余加工时间最长的工件的工序
4	LWR	选择剩余加工时间最短的工件的工序

考虑如下染色体[1 2 2 1 4 4 2 1 3]，其中 1 代表规则 SPT，2 代表 LPT，3 代表 MWR，4 代表 LWR。在初始化阶段，有

$$S_1=\{o_{111},\ o_{211},\ o_{312}\}$$

$$\phi_1^*=\min\{3,\ 1,\ 3\}=1$$

$$m^*=1$$

$$C_1=\{o_{111},\ o_{211}\}$$

工序 o_{111} 和 o_{211} 竞争机器 m_1。因为在给定染色体中的第一个基因是 1 (SPT 优先规则)，工序 211 被安排在机器 m_1，如图 6.10 (a)所示。更新后

$$S_2=\{o_{111},\ o_{223},\ o_{312}\}$$

$$\phi_2^* = \min\{4, 6, 3\} = 3$$
$$m^* = 2$$
$$C_2 = \{o_{312}\}$$

工序 o_{312} 被安排在机器 m_2，如图 6.10 (b)所示。

$$S_3 = \{o_{111}, o_{223}, o_{321}\}$$
$$\phi_3^* = \min\{4, 6, 3\} = 3$$
$$m^* = 1$$
$$C_3 = \{o_{111}, o_{321}\}$$

工序 o_{111} 和 o_{321} 竞争机器 m_1。因为给定染色体中的第三个基因是 2 (LPT 优先规则)，工序 o_{111} 被安排在机器 m_1，如图 6.10(c)所示。重复以上步骤直到得到一个完全调度，如图 6.10(d)所示。

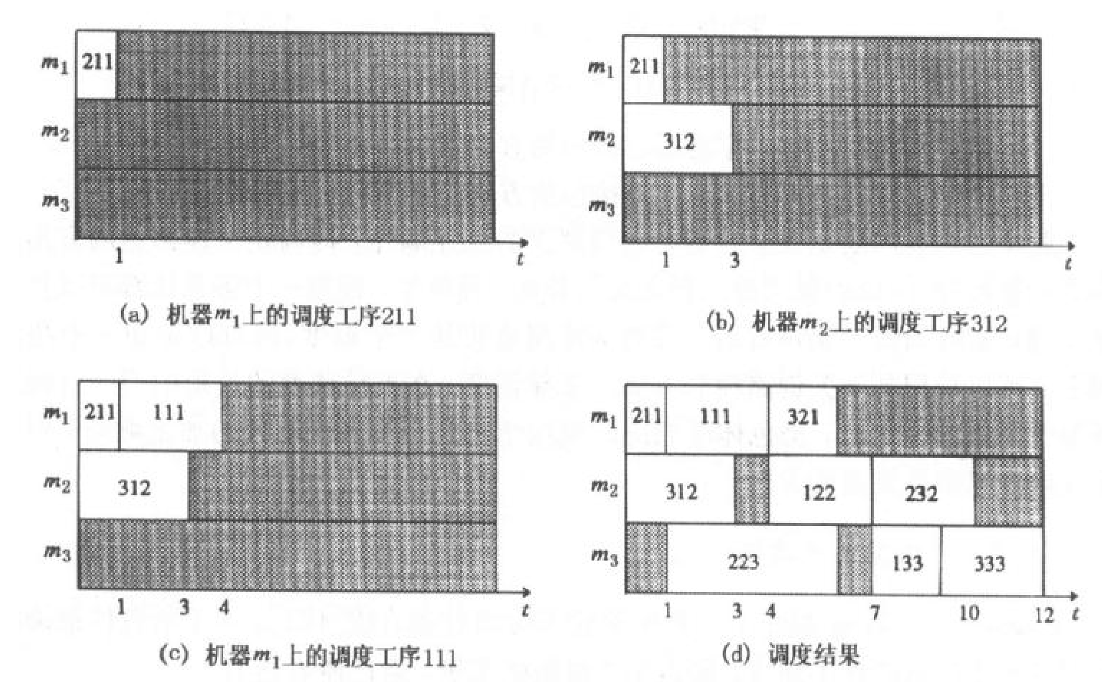


图 6.10 基于优先规则编码的推导调度

6. 基于非连接图表达法

Tamaki 和 Nishikawa 在文献[397]提出了一个基于非连接图的表达法，也可视为一种基于工件对关系的表达法。作业车间调度问题可以用一个非连接图表示[23,360]。非连接图 $G = (N, A, E)$ 定义为：N(节点) 是所有工序的集合，A 是连接同一工件的连续(紧邻)工序的所有边的集合，E 是连接在同一机器上加工的工序的非连接边的集合。非连接约束用 E 中的边表达。一条非连接边能够用其两个方向中的任意一个来标定。构造一个调度必须标定所有非连接边的方向，以确定同一机器上工序的顺序，一旦确定了一台机器的工序顺序，非连接边将被通常的连接边所代替。图 6.11 解释了一个 3 个工件、3 台机器的非连接图实例。一个染色体包含了一个对应 E 中非连接边有序列表的二进制串，如图 6.12 所示，其中 e_{ij} 代表节点 i 与 j 之间的非连接边，其定义为

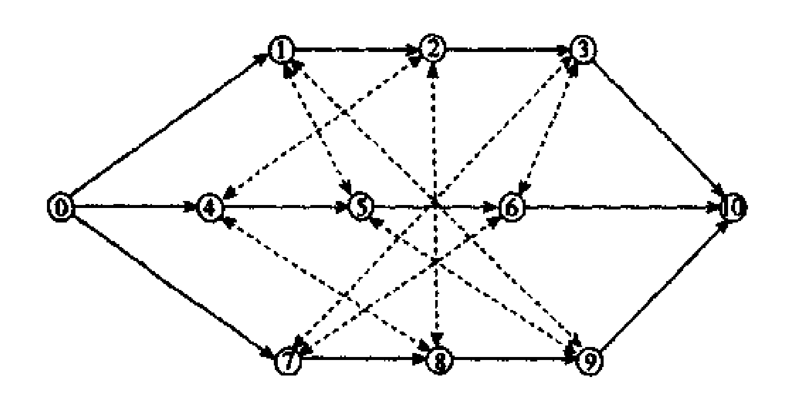


图 6.11　3 个工件、3 台机器的非连接图实例

非连接边有序列表	e_{15}	e_{19}	e_{59}	e_{24}	e_{28}	e_{48}	e_{36}	e_{37}	e_{67}
染色体	[0	0	1	1	0	0	0	1	1]

图 6.12　基于非连接图表达法

$$e_{ij}=\begin{cases}1, & \text{标定非连接边的方向是从节点 } j \text{ 到节点 } i \\ 0, & \text{标定非连接边的方向是从节点 } i \text{ 到节点 } j\end{cases}$$

作业车间调度问题就是要找到每台机器上的工序顺序，即确定非连接边的方向使非连接图是非循环图，以确保工序之间无先后冲突。很明显，任意一个染色体都可以产生一个循环图，这时调度是非可行的，因此不能用来表达一个调度，但可以用作一个决策优先顺序。可以采用基于关键路径的方法来推导调度。在推导调度的过程中当一台机器上出现两个节点的冲突时，染色体的相应位被用于决定两个工序之间的加工顺序，即确定两个节点之间非连接边的方向。

7. 基于完成时间的表达法

Yamada 和 Nakano 提出了一种基于完成时间的表达法[423]。一个染色体被编码为一个工序完成时间的有序列表。如表 6.2 给出的实例，染色体表达为

$$[c_{111}, c_{122}, c_{133}, c_{211}, c_{223}, c_{232}, c_{312}, c_{321}, c_{333}]$$

其中 c_{jir} 表示工件 j 在机器 r 上加工的第 i 道工序的完成时间。很容易看出这种表达法不实用于大多数的遗传算子，因为将产生非法调度。Yamada 和 Nakano 为此设计了一种专用的算子。

8. 基于机器的表达法

Dorndorf 和 Pesch 在文献[112]中提出了一种基于机器表达的遗传算法，染色体编码为机器的序列，根据该序列用瓶颈移动启发式构造调度。

由 Adams，Balas 和 Zawack[3]提出的瓶颈移动启发式可能是作业车间调度问题所有启发式算法中最有效的算法。算法中把机器一个一个地排列，每次在没有被调度的机器中识别一个瓶颈机器。当调度一个新的机器时以前的调度需要重新局部优化。瓶颈识别过程和重新局部优化是基于反复求解一个特定的单机调度问题来进行的。该问题是原问题的松弛问题。该方法的主要贡献在于用松弛来决定机器应该排列的顺序。

瓶颈移动启发式基于瓶颈机器优先的思想。机器瓶颈性能的不同度量方法将得到不同的瓶颈机器序列。用瓶颈移动启发式得到的调度性能依赖于瓶颈机器的顺序。Adams，Balas 和 Zawack 也提出了用隐枚举式的瓶颈移动启发式方法来考虑不同的机器顺序。

不同于隐枚举式的树搜索，Dorndorf 和 Pesch 提出了一种遗传策略：用瓶颈启发式算法确定最好的机器顺序。染色体是机器的有序列表，这里的遗传算法被用于进化染色体，并为瓶颈启发式算法找到一个好的机器顺序。两者不同之处在于选择下一个机器的决策不再由瓶颈决定，而是由染色体控制。

令 M_0 是已调度机器的集合，给定染色体为$[m_1, m_2, \cdots, m_m]$。从染色体推导出调度的过程如下：

基于机器编码的调度过程

步骤 1：令 $M_0 \leftarrow \{\phi\}, i \leftarrow 1$，染色体 为$[m_1, m_2, \cdots, m_m]$；

步骤 2：优化地排序机器 m_i，更新集合 $M_0 \leftarrow M_0 \cup \{m_i\}$；

步骤 3：重新依次优化每个关键机器 $m_i \in M_0$ 的顺序，保持其他的顺序不变；

步骤 4：令 $i \leftarrow i+1$，如果 $i > m$，停止；否则，返回步骤 2。

步骤 3 的详细过程参见 Adams，Balas 和 Zawack[3]或 Applegate and Cook[9]。

9. 随机键表达法

随机键表达法首先由 Bean[29]提出。采用这种技术遗传算子可以产生可行的后代，而且不会因大量的调度和优化问题而增加总的计算开销。Norman 和 Bean 进一步成功地把这种方法推广到了作业车间的调度问题[317,318]。

随机键表达法用随机数编码成一个解，其值用来调度键以便对解进行解码。对于 n 个工件、m 台机器的调度问题，每个基因(一个随机键)包含两部分，集合$\{1,2,\cdots,m\}$中的整数和由(0,1)间随机产生的分数。整数部分用来分配给工件的机器，分数部分则用来为每台机器上的工件排序。考虑表 6.2 给出的相同实例，假设染色体为

$$[1.34\ 1.09\ 1.88\ 2.66\ 2.91\ 2.01\ 3.23\ 3.21\ 3.44]$$

机器 1 按增加的顺序调度键，得到工件的顺序 2→1→3，对于机器 2，工件的顺序为 3→1→2，对于机器 3，工件的顺序为 2→1→3。令 o_{jm}表达机器 m 上的工件 j，染色体能转化为一个有序工序的唯一列表为

$$[o_{21}\ o_{11}\ o_{31}\ o_{32}\ o_{12}\ o_{22}\ o_{23}\ o_{13}\ o_{33}]$$

很容易看出以上给出的工件顺序可能违背先后约束。针对这种编码，Norman 和 Bean 给出了一种用于处理这种约束的伪编码[317]。

6.4.2 讨论

上一节讨论了各种编码技术，本节从以下几个方面作些比较：

(1) 染色体的 Lamark 性能；

(2) 解码器的复杂性；

(3) 编码空间的性能及映射；

(4) 内存需求。

1. Lamark 性能

由 Holland[220]提出并发展的简单遗传算法来源于达尔文的自然选择理论，具有很强的搜索能力，但搜索速度相对较慢。Kennedy[246,415]在简单遗传算法中引进了 Lamark 进化以改进其效率，其后代的有机体首先通过达尔文的生态进化，然后通过 Lamark 的智能进化，在评估前注入有机体一些“智能”。

编码技术的 Lamark 性能关心的是一个染色体能否通过一般的遗传算子把它的优点传递给下代种群。例如一个 9 城市的 TSP 问题能够编码成如下染色体：

随机键 [0.34 0.09 0.88 0.66 0.91 0.01 0.23 0.21 0.44]

序列 [6 2 8 7 1 9 4 3 5]

假设一条子路径由第 2 到第 6 个断点组成，对于顺序表达法，子路径为[8 7 1 9]，后代从其双亲得到相同的子路径。对于随机键表达法，子路径为[0.88 0.66 0.91 0.01]。通常地，在后代中可以得到不同的子路径，但子路径 8→7→1→9 与其双亲相同，得到什么子路径取决于后代中其他基因的值。事实上后代只能从其双亲中获得微小的随机变化。

一般地，我们希望编码技术具有 Lamark 性能。事实上大多数的编码都具有这种性能，也就是说，后代能从其双亲中继承一些好的性能。反之，没有 Lamark 性能，即后代不能从双亲中继承任何性能；第三种情况是半 Lamark 性能，即后代的一部分继承了双亲的性能，而其余的部分没有继承。

让我们考虑上一节给出的一些编码技术。随机键表达法和基于完成时间的表达法没有 Lamark 性能，基于工件表达法和基于优先规则表达法有 Lamark 性能，其它的表达法属于半 Lamark 性能。一般地，对于 n 个工件、m 台机器问题，一个染色体包含 $n\times m$ 个基因，每个基因对应于一道工序。一一对应关系按以下三种方式实现：①工件顺序，②优先列表，③优先规则。对于前面两种情况，相应的关系依赖于前后关系，意思是说具有相同值的基因可以解码成相同的工序，但也可以不相同。

2. 解码复杂性

原则上作业车间调度问题有无数个可行的调度，一般地可分为三种：①半活动调度，②活动调度，③非延迟调度。一个最优调度在活动调度集中，非延迟调度数少于活动调度数，但不能保证将包含一个最优解，因此，我们希望由染色体解码得到的调度是一个活动调度。作业车间调度问题的所有编码技术通过解码都能够得到活动调度．解码的复杂度可以分为以下四个级别：

0 级：无解码器。基于完成时间的表达法属于此类。在这种情况下，所有的任务都加在遗传算子上。

1 级：简单映射关系。基于工序表达法、基于工件对关系表达法和基于工件表达法属于此类。

2 级：简单启发式。基于优先列表表达法和基于优先规则表达法都属于此类。

3 级：复杂启发式。Kobayashi 等采用的基于优先列表表达法、基于非连接图表达法，基于机器表达法都属于此类。

3. 编码空间性能和映射

对于任何编码技术，通过解码过程，一个染色体对应一个合法的、可行的、活动的调度，映射关系是一一对应的。但编码空间可以分为两类：一类是包含可行解空间，另一类是包含非可行解空间。测试的方法是检测单点交叉算子产生的后代是否合法。基于优先规则表达法、基于工件表达法和基于非连接图表达法属于第一类，其他属于第二类。

另外，基于工件表达法和基于机器表达法的编码空间只是解空间的部分空间。

4. 内存需求

对于 n 个工件、m 台机器问题，如果我们定义染色体的标准长度为 $n \times m$ 个基因，作业车间问题的编码可以分为三类：

(1) 编码长度小于标准长度。基于工件的编码长度、基于非连接图的编码长度、基于机器的编码长度属于此类。值得注意的是基于工件的编码长度的空间只对应部分解空间，而不是全部解空间，因此不能保证最优。

(2) 编码长度大于标准长度。只有基于工件对关系的编码长度属于此类。这种表达法高度冗余。

(3) 编码长度等于标准长度。其他表达法属于此类。

最后，我们希望指出为了给出关于这些编码技术的公正评价，有必要对这些编码技术在标准实验条件下用基准问题的运行性能进行深入的比较，以揭示其优点和局限性。我们将如何设计这样的标准实验放在第 6.9 节。

6.4.3 混合式遗传搜索

组合优化问题是遗传算法求解的领域，与传统的启发式算法相比，遗传算法不适于邻域最优解的微调结构，因此把传统的启发式算法（如局域搜索）嵌入到遗传算法中构造一个更强的遗传算法是必不可少的。关于作业车间调度问题的各种混合式算法可以分为两类基本的方法[101,353]。

(1) 修改的遗传算子；

(2) 传统的启发式嵌入到遗传算法中。

第一类方法是试图基于传统启发式算法提出新的遗传算子，例如，①基于 Giffler 和 Thompson 算法[423]设计新的交叉算子，或②基于邻域搜索技术[71]设计新的变异算子。第二种方法是传统的启发式和遗传算法的结合，这可通过多种方式实现，主要包括：

(1) 把启发式嵌入到初始化中产生一个适应性好的初始解。按这种方式，混合式遗传算法能够保证优于传统启发式。

(2) 将启发式嵌入到评估函数中将染色体解码为调度。

(3) 局域搜索启发式嵌入到遗传算法的基本环中，同变异和交叉算子一起作用，在评估前对后代施行快捷且局部的优化。

采用混合式方法，遗传算法被用于个体中的全局搜索，而启发式在染色体中施行局

部探寻。由于遗传算法和启发式算法的互补性能，混合式算法往往优于任何单独的算法。

1. 基于 Giffler 和 Thompson 算法的交叉算子

Yamada 和 Nakano 提出了一种基于 Giffler 和 Thompson 算法的交叉算子[423]。Giffler 和 Thompson 算法的产生步骤是一种树搜索方法。在每一步，它实质上识别所有的加工冲突(竞争同一机器的工序)，然后用隐枚举方法重新解决所有冲突。Yamada 和 Nakano 交叉算子的步骤本质上是一类一次通过步骤，而不是树搜索方法。在产生后代的每一步，像 Giffler 和 Thompson 方法一样，识别所有加工的冲突，然后根据父代调度从工序的冲突集合中选择一道工序。令

o_{ji}——工件 j 的第 i 道工序；

S——对于一个给定的部分调度的可调度工序的集合；

ϕ_{ji}——S 中工件 j 的第 i 道工序的最早完成时间；

G_r——机器 r 上 S 中冲突工序的集合。

产生后代的步骤如下：

基于 Giffler 和 Thompson 算法的交叉算子过程

步骤 1：令 S 初始时是包含无紧前工序的所有工序。

步骤 2：确定 $\phi^* = \min\ \{\phi_{ji} \mid o_{ji} \in S\}$ 和能够实现 ϕ^* 的机器 r^*。

步骤 3：令 G_{r^*} 包含所有需求机器 r^* 的工序 $o_{ji} \in S$。

步骤 4：从 G_{r^*} 中选择一道工序：

1) 产生一个随机数 $\varepsilon \in [0,1)$，并与变异率 p_m 比较，如果 $\varepsilon < p_m$，那么从 G_{r^*} 中任意选择一道工序为 o_{ji}^*；

2) 否则以相同的概率任选一个双亲，如 p_s，从 G_{r^*} 的所有工序中找一个在 p_s 中调度最早的工序 o_{ji}^*；

3) 根据 ϕ_{ji} 在后代中调度 o_{ji}^*。

步骤 5：更新 S

1) 从 S 中删除工序 o_{ji}^*；

2) 在 S 中增加一个 o_{ji}^* 的直接后续工序。

步骤 6：返回步骤 2，直到产生一个完整的调度。

步骤 4 的 1)是随机地选择工序来消除冲突，而在步骤 4 的 2)中，是为 G_{r^*} 中所有冲突工序的双亲 p_s 中调度最早的工序赋给优先度来消除冲突。但双亲 p_s 是以相同的概率随机选取的。因此 Yamada 和 Nakano 方法本质上是基于优先分配启发式方法，而不是一个纯 Giffler 和 Thompson 方法。在每一步，选择工件加入到后代的部分调度中，工序之间的冲突是根据双亲调度给定的工序优先权来解决的。

图 6.13 标明步骤 4 的 2)中工序的选择。对于后代，工序 o_{11},o_{21}，o_{31}和 o_{32}已调度，可调度的工序为 $S=\{o_{12}, o_{22}, o_{33}\}$。假设冲突工序的集合为 $S=\{o_{12}, o_{22}\}$，选择双亲 p_1。因为工序 o_{12}在双亲 p_1 中最早调度，所以被调度进后代中。

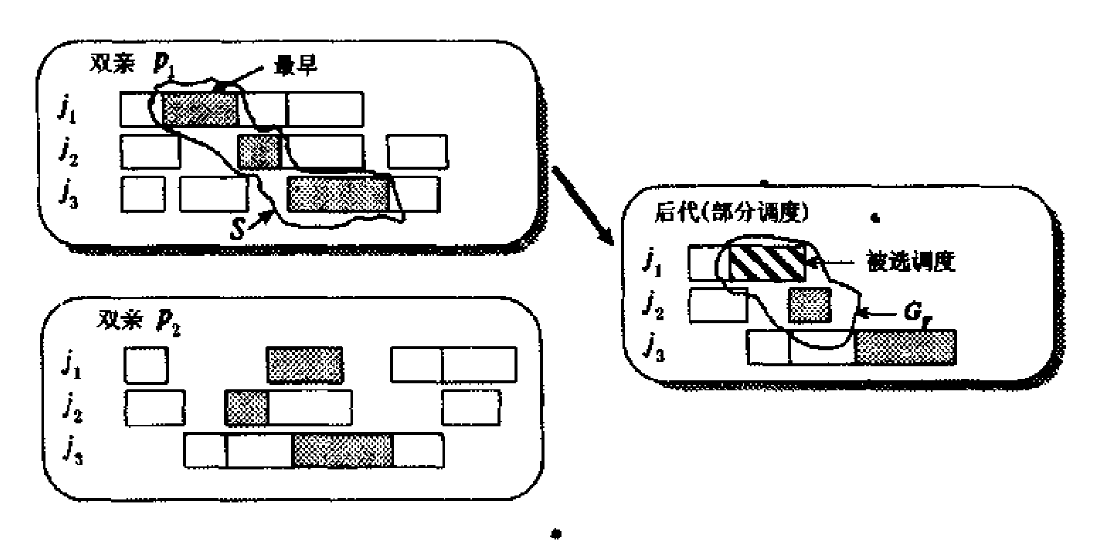


图 6.13 基于 Giffler 和 Thompson 算法的交叉

2. 基于邻域搜索的变异

在传统的遗传算法中，变异只是后备算子，用于产生关于染色体的微小扰动以维持个体的分散性。当设计一个混合式遗传算法时，基本的原则是什么地方可以混合就在什么地方混合。这里变异设计成邻域搜索，它不再是后备算子，而是用于施行深度搜索以便找到改进的后代。

关于调度的邻域，有多种定义。对于基于工序的表达法，给定染色体的调度邻域被视为由给定染色体通过交换 λ 基因(随机选择但不同的基因)的位置得到的染色体的集合。如果一个染色体按照某种度量比邻域中的任意一个染色体都好，那么就说该染色体是 λ 最优。让我们看 4 个工件 4 台机器问题的实例，假定随机选取位置 4,8 和 12 的基因为 (4 3 2)，可能的调度为 (3 4 2)，(3 2 4)，(2 3 4)，(2 4 3) 和 (4 2 3)。基因的可能调度及染色体的不变部分形成邻域染色体，如图 6.14 所示。然后评估所有邻域染色体，最好的染色体作为变异的后代。变异的总体步骤如下：

双亲染色体

2 4 1 4 2 3 1 3 4 2 3 2 1 3 1 4

邻域染色体

2 4 1 3 2 3 1 4 4 2 3 2 1 3 1 4
2 4 1 3 2 3 1 2 4 2 3 4 1 3 1 4
2 4 1 2 2 3 1 3 4 2 3 4 1 3 1 4
2 4 1 2 2 3 1 4 4 2 3 3 1 3 1 4
2 4 1 4 2 3 1 2 4 2 3 3 1 3 1 4

图 6.14 邻域染色体

基于邻域搜索的变异过程

begin

$i \leftarrow 0$

```
    while i ≤ pop_size × p_m do
        随机选择一个未变异的染色体；
        从中随机挑出λ个不同的基因；
        基于基因的所有排列构造邻域；
        评估所有邻域调度；
        挑选最好的邻域染色体作为后代；
        i←i+1；
    end
end
```

3. 与局部搜索相结合的遗传算法

混合式遗传算法的一种常见形式是遗传算法与局部搜索的结合。遗传算法善于全局搜索，但收敛速度慢，然而局部搜索善于微调但易陷入局部最优。混合式方法结合了这两种方法的性能，遗传算法用于全局搜索避免陷入局部最优，局部搜索用于引导微调。

本文中的局部搜索可以看作是个体在生命期间的一种学习。对于标准遗传算法，染色体的选择基于出生时的适应性，然而对于混合式算法，染色体的选择则基于个体在实行局部搜索期间最后时刻的适应性。改进的后代将学习过程中获得的性能通过公共交叉传递给后代，这种现象叫做 Lamark 进化。因此这种遗传算法可被视为达尔文进化与 Lamark 进化的结合。

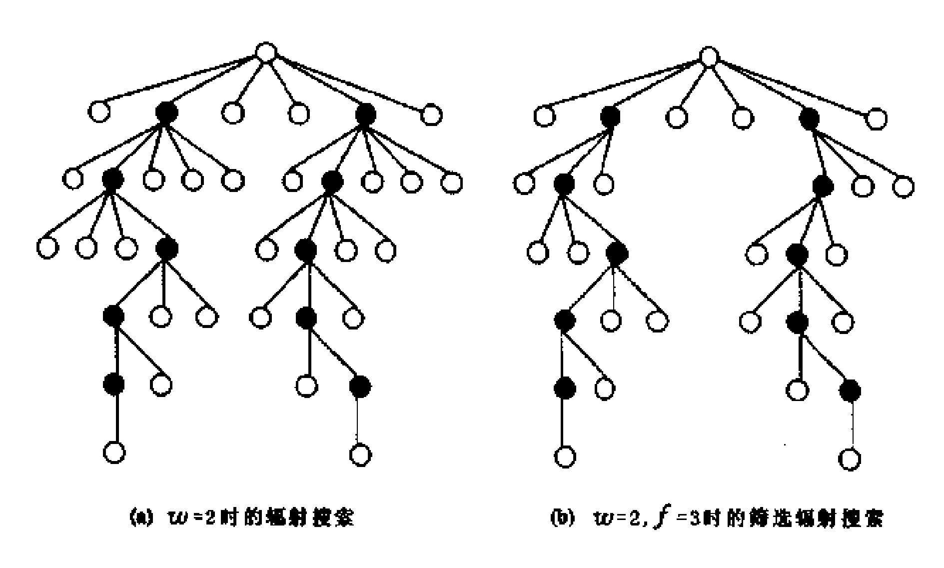
```
遗传算法 十局部搜索过程
begin
    t←0；
    初始化 P(t)；
    用局部搜索改进 P(t)；
    评估 P(t)；
    while 不满足终止条件 do
        重新组合 P(t)产生 C(t)；
        用局部搜索改进 C(t)；
        评估 C(t)；
        从 P(t)和 C(t)中选择 P(t+1)；
        t← t+1；
    end
end
```

4. 遗传算法与辐射搜索的结合

Holsapple，Jacob，Pakath 和 Zaveri 提出的混合式遗传算法是遗传算法与辐射搜索技术的结合[221]。粗略地说，辐射搜索(beam search)用于把一个染色体(基于工件表达法)转换为一个调度。它是广度优先搜索启发式的一种改进。广度优先搜索根据辐射宽度来限制搜索树的每阶段(层次)所需要扩展的节点数。在搜索过程中的每个层次，用预先

定义的评估函数来评估所有节点，然后用辐射宽度 w 来挑选 w 个最好的节点，并从现行层次扩展产生树中的下一层次的子节点，现行层次剩下的节点被永远剪切掉。如果开始时少于 w 个可用的节点，那么所有的节点均用来进行扩展；然后从被选用的 w 个节点的每个节点出发，产生所有可能的后继节点，整个过程持续到没有进一步的扩展为止。图 6.15(a)用一个简单的实例解释辐射搜索。

筛选辐射搜索是辐射搜索方法的一种改进，用另一种称为筛选宽度(filter width)的结构来进一步剪切状态空间。对于每个特定的层次被选作扩展的每个节点，筛选的宽度 f 决定了被产生的后继节点的最大个数，也就是说能够产生不超过 f 个的尽可能多的节点。对于一个给定的值 f，后继节点的选择是随机的、但 w 和 f 是用户事先确定的值。搜索过程如图 6.15(b)所示。



(a) w=2时的辐射搜索　　(b) w=2, f=3时的筛选辐射搜索

图 6.15　辐射搜索与筛选辐射搜索

Holsapple 等人采用基于工件的表达法，用遗传算法计算工件的顺序，用筛选辐射搜索产生给定工件顺序下的最好调度。这里可以把辐射搜索考虑成评估函数的一部分，即把工件顺序转换成一个调度，但又不仅仅是转换。Holsapple 等人讨论了与柔性制造相关的一种静态调度问题。该问题与古典作业车间调度问题的区别是工件的任何工序可以在任何机器上加工。在这种情况下，基于工件表达法的染色体对应于多个可行调度，因此用辐射搜索来确定给定染色体的可行调度中的最好调度。对于古典作业车间调度问题，每道工序的加工机器是预先确定的，因此基于工件的染色体如上节所述只对应一个可行调度，在这种情况下不需要用辐射搜索那样的任何树搜索技术来转换染色体为可行调度。混合式遗传算法的总体步骤为

遗传算法 +辐射搜索过程

begin

$t \leftarrow 0$;

初始化 $P(t)$(工件顺序)；

对 $P(t)$中的每个染色体施行辐射搜索产生最好的调度；

对于 $P(t)$评估调度，确定适值；

```
  while 不满足终止条件 do
    重新组合 P(t)产生 C(t);
    对于C(t)中的每个染色体施行辐射搜索产生最好的调度;
    对于C(t)评估调度，确定适值;
    从 P(t)和 C(t)中选择下一代种群 P(t+1);
    t← t+1;
  end
end
```

6.5 Gen-Tsujimura-Kubota 方法

Gen，Tsujimura 和 Kubota 提出了一种用遗传算法求解作业车间调度问题的实现方法[165]，提出了基于工件的表达法，并设计了一个部分调度交换交叉算子。在部分调度交换交叉算子中，考虑部分调度为自然构成块(natural building blocks)，以类似于 Holland[220]描述的方式，试图用这样的交叉使后代保留这些构成块。部分调度用调度中起始和最终位置相同的工件来识别。例如，有两个双亲染色体 p_1 和 p_2，如图 6.16 所示，随机选择部分调度如下：

(1) 在双亲 p_1 中随机选择一个位置。假设是第 6 个位置，对应工件 4。

(2) 在同一双亲 p_1 中找出最邻近工件 4 的位置 9，这样就得到部分调度 1(*partial schedulie*$_1$)为[4 1 2 4]。

(3) 下一个部分调度不是由 p_2 随机产生的，如同第一个部分调度，必须由工件 4 起始和终止。值得注意的是，两个工件 4 是 p_1 中的第一个和第二个工件 4。这样，部分调度 2(*partial schedule*$_2$)由 p_2 中的第一和第二个工件 4 之间的基因组成，即为[4 1 3 1 1 3 4]。

双亲染色体：

部分调度1

p_1 [3 2 1 2 3 4 1 2 4 4 1 3 4 1 2 3]

p_2 [4 1 3 1 1 3 4 1 2 3 4 2 2 2 3 4]

部分调度2

图 6.16 选择部分调度

交换部分调度如图 6.17 所示。部分调度包含不同的基因个数，因此通过交换产生的后代可能是非法的。后代中丢失和过剩的基因由图 6.18 来确定。

交换部分调度：

部分调度2

o_1 [3 2 1 2 3 4 1 3 1 1 3 4 4 1 3 4 1 2 3]

o_2 [4 1 2 4 1 2 3 4 2 2 2 3 4]

部分调度1

图 6.17 交换部分调度

o_1 丢失和剩余的基因:

部分调度 1　4 1 2 4　—丢失的基因→　2

部分调度 2　4 1 3 1 1 3 4　—剩余的基因→　3 1 1 3

o_2 丢失和剩余的基因:

部分调度 2　4 1 3 1 1 3 4　—丢失的基因→　3 1 1 3

部分调度 1　4 1 2 4　—剩余的基因→　2

图 6.18　丢失和过剩的基因

下一步通过删除过剩的基因和增加丢失的基因，使后代合法化。对于后代 o_1，过剩的基因是[3 1 1 3]。后代 o_1 从 p_2 中得到部分调度 2。因为在部分调度 2 中的工件 3 和工件 1 是 p_2 中的第一个工件 3 和第一个工件 1，必须删除 o_1 中部分调度前的工件 3 和工件 1，以便与 p_2 中的顺序保持一致。原因在于基于工序表达法中的基因对应与之关联的工序。如果 o_1 中部分调度 2 的基因与 p_2 中基因的顺序一致，这些基因将在两种情况下对应相同的工序，否则对应不同的工序。我们希望后代 o_1 继承双亲 p_2 中的部分调度 2 标识相同的部分调度，使部分调度 2 中的基因在 o_1 和 p_2 中有相同的顺序。对于后代 o_1，其丢失的基因是工件 2。因为部分调度 2 中没有工件 2，因此将工件 2 插入到除部分调度 2 的任何位置，并不改变部分调度 2 中基因的顺序。后代的合法化过程如图 6.19 所示。

合法化后代 o_1:

(1) 删除剩余的基因 3 1 1 3

o_1 [[illegible] 2 [illegible] 2 [illegible] 4 1 3 1 1 3 4 4 [illegible] 3 4 1 2 3]

↓

o_1 [2 2 4 1 3 1 1 3 4 4 3 4 1 2 3]

(2) 插入丢失的基因 2

o_1 [2 2 4 1 3 1 1 3 4 2 4 3 4 1 2 3]

图 6.19　后代 o_1 的合法化

对于后代 o_2，其过剩的基因是工件 2。我们可以删除 o_2 中除部分调度 1 外的任意工件 2，其丢失的基因是 [3 1 1 3]。我们也必须在后代 o_2 中的部分调度 1 前和后各插入一个工件 1，使得部分调度 1 中的工件 1 为后代 o_2 中的第二个工件 1。因为部分调度 1 中没有工件 3，插入工件 3 到(除部分调度 1 中的位置)任何位置，如图 6.20 所示。

Gen，Tsujimura 和 Kubota 用工件对交换变异，即随机选取两个不完全一致的工件交换位置，如图 6.21 所示。对于基于工序的表达法，染色体和调度之间的映像关系是多对一，通过交换两个相邻的工件得到的后代可以产生与双亲相同的调度，因此两个不完全相同的工件之间的间距越大，效果越好。

此外，Tsujimura，Gen 和 Kubota 还把他们的工作扩展到了工件的加工时间是三角模糊数的模糊作业车间调度的问题中，用于处理加工时间的非确定性[406]。

合法化后代 o_2:

(1) 删除剩余的基因 2

o_2 [4 1 2 4 1 2 3 4 2 2 2 3 4]

↓

o_2 [4 1 2 4 1 3 4 2 2 2 3 4]

(2) 插入丢失的基因 3 1 1 3

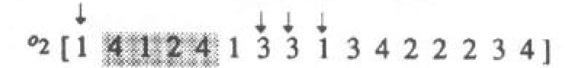


图 6.20 后代 o_2 的合法化

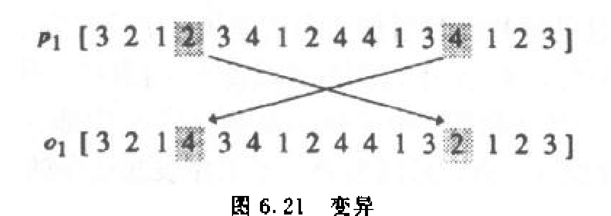


图 6.21 变异

6.6 Cheng-Gen-Tsujimura 方法

Cheng, Gen 和 Tsujimura 进一步改进了 Gen, Tsujimura 和 Kubota 的方法，以提高其有效性。主要改进包括以下三个方面：

(1) 设计一个新的解码步骤确保产生活动调度；

(2) 提出一种简化的交叉算子；

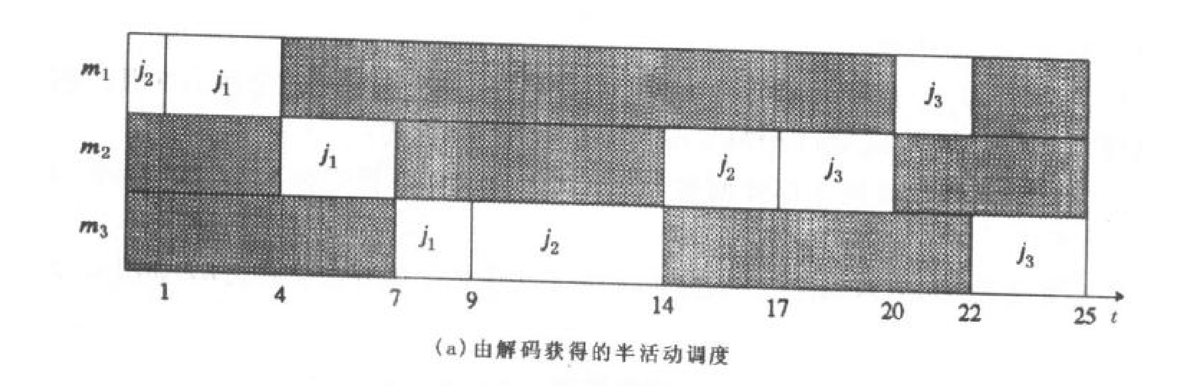
(3) 用邻域搜索技术设计变异，以便进一步地搜索改进的后代，详见第 6.4.3 节。

在 Gen，Tsujimura 和 Kubota 的方法中，当染色体解码产生调度时，他们仅考虑两件事情：给定染色体中工序顺序和工件的工序先后约束。他们同时还在调度中确定了每台机器上工序的顺序。解码步骤只能确保产生半活动调度，而不能确保产生活动调度，就是说，在调度中只存在允许左移(permissible left-shift)或全局左移(global left-shift)。因为最优解是一个活动调度，解码步骤影响了本算法对于大规模作业车间问题的有效性。

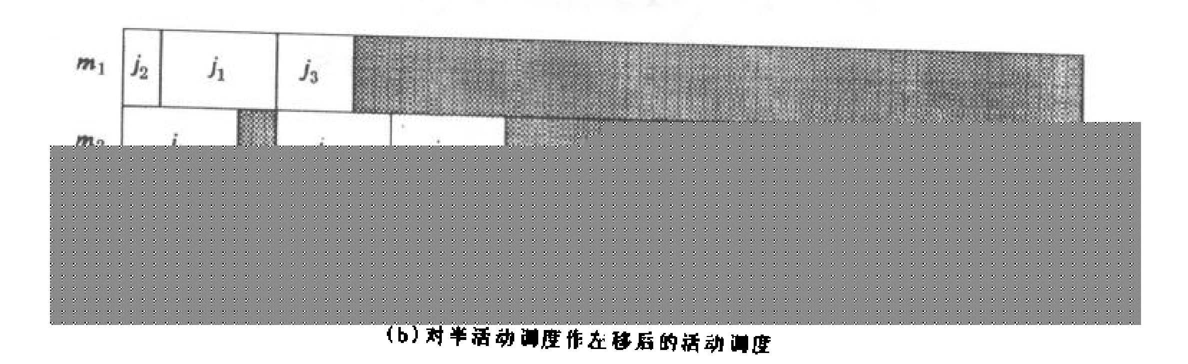
现在我们看一个实例。仍用表 6.2 给出的 3 个工件、3 台机器的问题。假设染色体为 [2 1 1 1 2 2 3 3 3]。根据解码步骤，我们可以得到相应的机器列表[1 1 2 3 3 2 2 1 3]。根据机器列表，可以作一个调度，如表 6.6 所示，它确定了每台机器上工序的顺序。

表 6.6 调度

机器	工件顺序		
m_1	j_2	j_1	j_3
m_2	j_1	j_2	j_3
m_3	j_1	j_2	j_3



(a) 由解码获得的半活动调度



(b) 对半活动调度作左移后的活动调度

图 6.22　左移在改变半活动调度中的影响

图 6.22(a) 表示对于给定的染色体通过解码步骤得到的半活动调度的甘特(Gantt)图。该半活动调度的流程时间为 25，且有两个允许左移：

(1) 工件 j_3 在机器 m_2 上可以在时间 0 开始加工；

(2) 工件 j_2 在机器 m_3 上可以在时间 1 开始加工。

通过施行这些左移，可以得到一个活动调度，如图 6.22(b)所示，流程时间为 12。

Cheng，Gen 和 Tsujimura 改进了解码步骤，确保由一个给定染色体产生一个活动调度，对于基于工序表达法，每个基因唯一地对应一道工序，改进的解码步骤首先把染色体解码为有序的工序列表；基于列表法则根据一次通过启发式产生一个调度，首先排列表中的第一道工序，然后考虑第二道工序，如此进行。每道处理中的工序的加工时间位于工序要求的机器的最可能得到的加工时间，这个过程重复进行直到列表中所有的工序都被排到适当的位置。通过改进步骤得到的调度确保是一个活动调度，这里的染色体可以被视为分配了一个优先权给每道工序。列表中从左到右的位置，被标为从最小到最大；一个在列表中处于较低位置的工序有较高的优先权，可比其他有较高位置的工序优先调度。

令 o_{jim}表示工件 j 在机器 m 上的第 i 道工序，染色体 [2 1 1 1 2 2 3 3 3]可被解码为有序工序的唯一列表[o_{211} o_{111} o_{122} o_{133} o_{223} o_{232} o_{312} o_{321} o_{333}]。工序 o_{211}有最高的优先权，首先调度，然后调度 o_{111}。以此类推，最后得到的活动调度如图 6.22(b)所示。

上节给出的交叉算子相当复杂，因为它还要求后代中部分调度的工序顺序与他们在双亲中的保持一致，改进的方法用染色体来确定每道工序的优先权，然后根据一次通过优先分配启发式产生调度，因此不要求双亲和后代中部分调度的工序顺序保持一致。部分

调度交换算子可简化为

(1) 首先，从 p_1 和 p_2 中分别选出两个部分调度，它们包含相同个数的基因。交换部分调度产生后代，图 6.23(a) 给出了由这种方法产生的后代 o。

(2) 然后通过比较两个部分调度，找出对于后代 o 的丢失基因和过剩基因，如图 6.23(b)所示。

(3) 通过以随机方式删除过剩的基因和增加丢失的基因合法化后代 o，如图 6.23(c)所示。

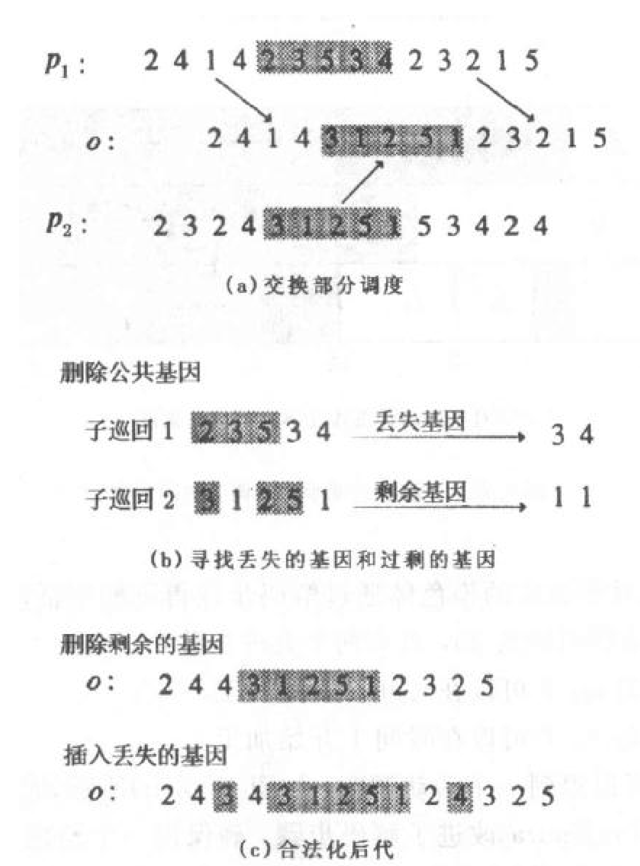


图 6.23 交叉

6.7 Falkenauer-Bouffouix 方法

Falkenauer 和 Bouffouix 采用基于优先列表的表达法，提出了具有下达时间和交货期的作业车间调度问题的遗传算法的实现方法[122]；之后，Croce，Tadei 和 Volta 进一步改进了 Falkenauer-Bouffouix 方法，用于处理古典作业车间调度问题[86]。

对于 n 个工件、m 台机器的作业车间调度问题，优先列表编码的染色体由 m 个子染色体组成。包含 n 个基因的每个子染色体，是对于一台机器的工序的优先列表。实际的调度可基于工序的优先列表，通过模拟由染色体解码得到。

采用这种染色体，遗传搜索的目的是找出列表中工序的最好顺序。由于从染色体推导调度的过程中，工序的优先权是根据工序在列表中的相对位置确定的，因此在执行交叉算子时应尽可能多地保持基因之间的相对位置。为此，Falkenauer 和 Bouffouix 提出了

一个改进的顺序交叉算子(OX)。Davis[99]首先讨论过OX，他倾向于转换基因之间的相对位置，而不是绝对位置。由于OX是为TSP问题设计的，所以在OX中，染色体被视为环状。在作业车间问题中，染色体不再是环状。为此，提出了OX的一种变形，即线性顺序交叉(LOX)，染色体是线性结构而不是环状结构。LOX独立地应用于每个子染色体。为便于解释，用子染色体为双亲。LOX按如下进行：

(1) 从双亲中随机选择子列表。如图6.24(a)所示。

(2) 从 p_1 中删除子列表2，留下一些"洞"(holes)，用h标识；然后从极端位置向中间滑动"洞"，直到达到交叉区域，如图6.24(b)所示。类似地，从 p_2 中删除子列表1，滑动"洞"直到交叉区域，如图6.24(c)的所示。

(3) 在 p_2 的"洞"中插入子列表1，形成后代 o_1，在 p_1 的"洞"中插入子列表2形成后代 o_2，如图6.24(d)所示。

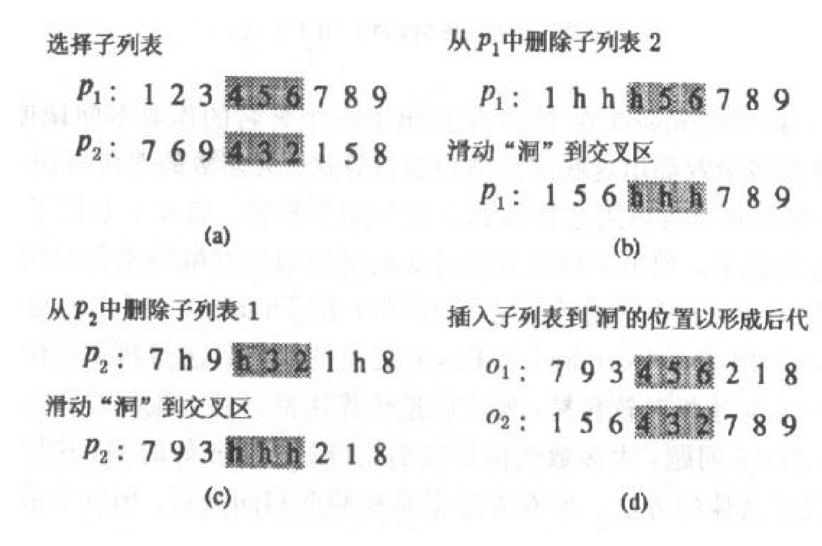


图6.24 线性顺序交叉LOX

交叉算子尽可能多地保持基因之间的相对位置和相应的双亲两端的绝对位置，两端对应于优先权最高和最低的工序。

Falkenauer和Bouffouix用反转作为变异算子；Croce，Tadei和Volta用基因之间的交换作为变异算子。

6.8 Dorndorf-Pesch方法

Dorndorf和Pesch提出了遗传算法用于作业车间调度问题的两种实现方法：基于优先规则的表达法和基于机器的表达法[112]。两种方法的共同特性是把遗传算法与传统启发式算法结合了起来。

对于基于优先规则的表达法，他们把著名的Giffler和Thompson算法嵌入到遗传算法中；遗传算法仅用于进化优先分配规则序列，而Giffler和Thompson算法则用于从优先分配规则编码中推导出调度。因为Giffler和Thompson算法是一种树形结构方法，事实上他们用的是一类一次通过启发式即优先分配启发式方法，因此不是纯粹的Giffler和Thompson算法[21]。

对于基于机器的表达法，他们把著名的瓶颈移位启发式嵌入到遗传算法中。遗传算法用于进化机器调度，瓶颈移位启发式则用于从机器调度编码推导出调度。事实上，当由一个给定的染色体推导一个调度时，他们仅用了瓶颈移位启发式的基本思想，即反复求解单机调度问题直到所有的机器都已经调度完为止。在每步迭代中，瓶颈移位启发式识别一个瓶颈机器并进行调度，然后识别下一个瓶颈机器，以此类推。在遗传算法中，机器的序列由给定染色体确定，因此没有必要在每一步都识别一个瓶颈机器。

因为这两种编码方法都可看作是一种换位表达法，像旅行商问题一样，可以应用许多传统的遗传算子。他们用标定窗口技术(scaling window techniques)来计算染色体的适值，以增加选择压力[206]，并用精英选择策略使得每一步中的最好解总能保留下来[171]。

6.9 计算结果与讨论

自从 Fisher 和 Thompson 在 1963 年提出了三个著名的作业车间调度问题[126]以来，运筹学界的许多学者都用这些问题来检验新算法。大多数的遗传算法-作业车间调度问题(GA-JSP)的研究人员也用这些问题来检验遗传算法。表 6.7 总结了一些遗传算法对这些调度问题的结果，列出了每种方法对这些问题的最好解的流程时间和最优解的流程时间，其中 Dorndorf1 表示遗传算法与 Giffler 和 Thompson 提出的启发式算法的混合式方法，Dorndorf2 表示 Dorndorf 和 Pesch 提出的遗传算法与瓶颈移位启发式算法的混合式方法。6×6 问题相对较容易，所有的遗传算法都得到了流程时间为 55 的最优解；对于 10×10 和 20×5 问题，大多数遗传算法都能得到相对较好的解，这样就很难从表中判断哪种方法优于其他的方法。所有方法都是检验的相同问题，但其实验条件不同。相同的实验条件是指

(1) 相同的染色体受检数；

(2) 相同的随机运行次数；

(3) 相同的软件和硬件计算设备。

表 6.7 Fisher 和 Thompson 的测试问题

	6×6	10×10	20×5
Optimal	55	930	1 165
Nakano 等 (1991)	55	965	1 215
Yamada 等 (1992)	55	930	1 184
Paredis 等 (1992)	—	1 006	—
Gen 等 (1994)	55	962	1 175
Fang 等 (1993)	—	949	1 189
Dorndorf1 等 (1995)	55	960	1 249
Dorndorf2 等 (1995)	55	938	1 178
Croce 等 (1995)	55	946	1 178
Cheng 等 (1995)	55	948	1 196

对于受检染色体的总数，在保持总数不变的情况下，可以取遗传算法的相同参数或不同参数；在相同的实验条件下，可以从下列五个方面判断所提出遗传算法的性能：

(1) 最好解 ；

(2) 最好解与最优解的偏差；

(3) 达到最好解的频率；

(4) 解的分布与统计分析；

(5) 内存与计算时间要求。

在标准实验条件下，对这些遗传算法用测试问题和其他大规模的问题作进一步的比较研究，有利于发现每个算法的优点和局限性；

表 6.8 给出了这些算法对 10×10 问题在某些参数下的结果。从表中可以看到在遗传算法的参数设置上有两种不同的选择，Nakano 和 Yamada 喜欢用大的种群数和小的迭代数，而 Gen 和 Cheng 却喜欢用相对小的种群数和较大的迭代步数。

表 6.8 GA 参数 (10×10)

	pop_size	*max_gen*	p_c	p_m
Nakano 等 (1991)	1 000	150	—	—
Yamada 等 (1992)	2 000	100	—	—
Gen 等 (1994)	60	5 000	0.4	0.3
Dorndorf1 等 (1995)	200	—	0.65	0.001
Dorndorf2 等 (1995)	40	—	0.65	—
Croce 等 (1995)	300	2971	1	0.03
Cheng 等 (1995)	40	2000	0.4	0.4

另一重要方面是遗传算法与传统启发式的比较。例如，对于 10×10 问题，Yamada 和 Nakano 在每次运行中检验了 2 100 个染色体[424]，也就是说，他们运行了 Giffler 和 Thompson 启发式算法的约 2 100 次。在 300 次实验中，有 5 次达到流程时间为 930 的最优解。对于相同的问题，Dorndorf 和 Pesch 用第一种混合式遗传算法检验了250 000个染色体，也就是说他们运行优先分配启发式几乎达 250 000 次。用第二种混合式遗传算法检验了 800 个染色体，即运行瓶颈移位启发式几乎达 800 次。问题是如果我们独立地运行相同次数的传统启发式能否得到一个更好的解。

直到现在、大多数遗传算法-作业车间调度问题的研究人员都把注意力转向到流程时间的度量上，因为遗传算法为进化计算提供了一个柔性的框架并且能够处理任何类型的目标函数和约束，能够用于处理具有非规则度量或多目标的非常复杂的问题，而这就是值得进一步研究的潜在领域。

对于遗传算法-作业车间调度问题的研究，为处理带约束的组合优化问题提供了丰富的经验。所有针对作业车间调度问题的技术可能也有助于现代柔性制造系统中的其他调度问题，如流水车间调度问题、混合式车间调度问题、动态作业车间的调度问题，以及其他的组合优化问题。

第七章 机器调度问题

7.1 引　　言

机器调度问题既有丰富的研究内容，同时又是一个在机械制造、逻辑、计算机结构等方面有广泛应用前景的研究领域。自从 Graham 等人[182]和 Graves[183]发表了机器调度问题的综述性文章以来，机器调度问题的文献迅速增加。机器调度问题的主要方面有：

(1) 机器配置；

(2) 工件特征；

(3) 目标函数。

机器配置通常可以分为单机和多机问题。Gupta 和 Kyparisis 概括地研究了单机调度问题[197]，Cheng 和 Sin 概括地研究了并行多机调度问题[74]。工件特征包括(但不局限于)工件之间的先后关系，工件下达时间，工件交货期和工件的优先权。Sen 和 Gupta 在文献[376]中综述了关于交货期的研究，Cheng 和 Gupta[73] 综述了把工件交货期作为决策变量的研究。目标函数包括单目标问题和多目标问题。Dileepan 和 Sen 研究了单机情况下的双目标调度问题[110]。目标函数又进一步分为规则和非规则度量。非规则性能度量可以随工件完成时间的减少而增加。Baker 和 Scudder[22]综述了关于非规则性能度量的研究。Koulamas[251]研究了总延迟时间问题。

机器调度问题是组合优化问题，其复杂性常常通过转换为已知复杂性的其他问题来确定。Lenstra，Rinnooy Kan 和 Brucker 综述性地研究了机器调度问题的复杂性[270]。

7.1.1 单机调度问题

基本的单机调度问题的假设可以描述为

(1) 一组由 n 个独立的单道工序工件在时间 0 等待加工；

(2) 工件的装设时间与工件的顺序无关，可以包括在加工时间内；

(3) 工件是可知的；

(4) 机器连续可用，在工件等待加工时机器不能闲置；

(5) 不允许工件预定机器。

问题的目标是确定工件的加工顺序使得一些性能的度量最优。

令$(j_1, j_2, \cdots, j_n)$表示工件，p_i 表示工件 j_i 的加工时间，d_i 表示交货期，w_i 表示权重，c_i 表示完工时间，r_i 表示准备时间。

流程时间(flowtime)F_i：定义为 $F_i = c_i - r_i$，即工件 j_i 在系统中的总逗留时间。流程时间用于度量系统对各个服务需求的反应，表示工件在到达和离开系统的时间长度。

延迟时间(lateness)L_i：定义为 $L_i = c_i - d_i$，即工件 j_i 的完工时间超过其交货期的总时间。延迟时间用于度量一个调度对于给定交货期的一致性，要么是正值，要么是负值。负延迟时间表示工件在交货期前完工。在很多情况下，伴随正延迟时间有惩罚和费用，

而负延迟时间没有多少价值。因此只需考虑正的延迟时间，称为拖期时间(tardiness)。

拖期时间 T_i：定义为 $T_i = \max\{0, L_i\}$，表示工件 j_i 不满足交货期的延迟时间，如果满足交货期，其值为 0。

令 Π 表示工件之间无闲置时间的可行调度的集合。对于一个给定的调度 $\sigma \in \Pi$，令 $f(\sigma)$ 表示相应的目标函数值，且 $r_i = 0$。

1. 平均流程时间问题

平均流程时间问题定义为

$$\min_{\sigma \in \Pi} f(\sigma) = \frac{1}{n} \sum_{i=1}^{n} F_i = \frac{1}{n} \sum_{i=1}^{n} c_i \tag{7.1}$$

可以用工件加工时间的非减顺序来调度工件，即 $p_{i_1} \leqslant p_{i_2} \leqslant \cdots \leqslant p_{i_n}$。这就是著名的最短加工时间调度规则(SPT 规则)。

2. 加权流程时间问题

加权流程时间问题定义为

$$\min_{\sigma \in \Pi} f(\sigma) = \sum_{i=1}^{n} w_i F_i = \sum_{i=1}^{n} w_i c_i \tag{7.2}$$

可以用 p_i/w_i 的非减顺序来调度工件，即 $p_{i_1}/w_{i_1} \leqslant p_{i_2}/w_{i_2} \leqslant \cdots \leqslant p_{i_n}/w_{i_n}$。这就是加权最短加工时间规则(WSPT)。

3. 平均拖期时间问题

平均拖期时间问题描述为

$$\min_{\sigma \in \Pi} f(\sigma) = \frac{1}{n} \sum_{i=1}^{n} T_i \tag{7.3}$$

4. 最大流程时间问题

最大流程时间问题定义为

$$\min_{\sigma \in \Pi} f(\sigma) = \max_{1 \leqslant i \leqslant n} \{F_i\} = \max_{1 \leqslant i \leqslant n} \{c_i\} \tag{7.4}$$

也称为制造周期(makespan)问题。

5. 最大延迟时间问题

最大延迟时间问题定义为

$$\min_{\sigma \in \Pi} f(\sigma) = \max_{1 \leqslant i \leqslant n} \{L_i\} \tag{7.5}$$

可以用最早交货期规则(EDD 规则)来调度工件，即 $d_{i_1} \leqslant d_{i_2} \leqslant \cdots \leqslant d_{i_n}$。

6. 最大拖期时间问题

最大拖期时间问题定义为

$$\min_{\sigma \in \Pi} f(\sigma) = \max_{1 \leqslant i \leqslant n} \{T_i\} \tag{7.6}$$

可以用 EDD 规则求解。

7. 最少拖期工件数问题

最少拖期工件数问题定义为

$$\min_{\sigma\in\Pi} f(\sigma) = \sum_{i=1}^{n} \delta(T_i) \tag{7.7}$$

其中

$$\delta(T_i) = \begin{cases} 1, & \text{如果 } T_i > 0 \\ 0, & \text{其他} \end{cases}$$

以上讨论的所有指标都是工件完工时间的函数，所以有如下一般形式：

$$\min_{\sigma\in\Pi} f(\sigma) = f(c_1, c_2, \cdots, c_n)$$

更进一步，它们都属于一类重要的性能度量——规则度量。如果满足

(1) 调度的目标是极小化 $f(\sigma)$；

(2) $f(\sigma)$增加，当且仅当至少有一个完工时间增加。

则一个性能度量是规则的。

8. 完工时间方差问题

完工时间方差问题 CTV (Completion Time Variance Problem) 定义为[286]

$$\min_{\sigma\in\Pi} f(\sigma) = \frac{1}{n} \sum_{j=1}^{n} (c_j - \bar{c})^2 \tag{7.8}$$

其中 $\bar{c}$ 是平均流程时间，由下式给出：

$$\bar{c} = \frac{1}{n} \sum_{j=1}^{n} c_j$$

在最优的 CTV 调度中，最长时间的工件首先加工[371]。

Gupta 等[196]研究提出的完工时间方差问题的权重形式(WCTV 问题) 定义为

$$\min_{\sigma\in\Pi} f(\sigma) = \frac{1}{n} \sum_{j=1}^{n} w_j (c_j - \bar{c})^2 \tag{7.9}$$

Cheng 和 Cai 已经证明 CTV 问题是 NP 难题，不存在求解 CTV 问题最优解的多项式或伪多项式算法[72]。

9. 等待时间方差问题

令 w_i 表示工件 j_i 的等待时间，$w_i = c_i - p_i$。等待时间方差问题 WTV (Waiting Time Variance Problem) 可以定义为

$$\min_{\sigma\in\Pi} f(\sigma) = \frac{1}{n} \sum_{j=1}^{n} (w_j - \bar{w})^2 \tag{7.10}$$

其中 $\bar{w}$ 是平均等待时间，由下式给出：

$$\bar{w} = \frac{1}{n} \sum_{j=1}^{n} w_j$$

最优 WTV 调度是 V 型调度，也就是说最短加工时间工件的紧前和后继工件分别按 LPT 和 SPT 顺序调度[116]。

10. 完工时间绝对偏差和问题

完工时间绝对偏差和问题(TADC)定义为

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \sum_{i=1}^{n}\sum_{j=i}^{n} | c_j - c_i | \tag{7.11}$$

$$= \sum_{i=1}^{n} (i-1)\ (n-i+1)\ p_{[i]} \tag{7.12}$$

其中[i]表示第 i 个加工的工件。

因为 CTV 问题的复杂性，Kanet 提出了 TADC 作为方差的度量，并且提出了一个求解该问题的 $O(n\log n)$算法[240]。TADC 和 CTV 问题的关键区别是前者为绝对偏差和后者为方差和。根据 Hardy，Littlewood 和 Polya 的方法，TADC 通过匹配位置权重$(i-1)\ (n-i+1)$的非增顺序与加工时间 $p_{[i]}$的非减顺序来求解[208,240]。

11. 等待时间的绝对偏差和问题

等待时间的绝对偏差和问题(TADW)可以描述为

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \sum_{i=1}^{n}\sum_{j=i}^{n} | w_j - w_i | \tag{7.13}$$

$$= \sum_{i=1}^{n} i\ (n-i)\ p_{[i]} \tag{7.14}$$

TADW 和 WTV 之间的关系类似于 TADC 和 CTV 之间的关系。TADW 可通过匹配位置权重 $i(n-i)$的非增顺序与加工时间 $p_{[i]}$的非减顺序来求解[16]。

以上给出的这些度量都是非规则度量。

7.1.2 提前/拖期(E/T)调度问题

调度模型中 E/T 惩罚的研究是最近兴起的一个研究领域[22]。多年来，这项研究主要侧重于对工件完工时间的非减规则度量，如平均流程时间，平均延迟时间，工件拖期率及平均拖期时间这样的度量。特别地，平均拖期时间准则尽管忽略了工件可能提前完工的结果，但它仍是度量交货期一致性的标准方式。然而，人们对准时生产(JIT)的兴趣改变了这种状态。准时生产源于提前和拖期都不应该鼓励的思想；E/T 惩罚概念的引进已经成为调度领域中一个新的迅速发展的研究分支。由于 E/T 惩罚的使用引出了一个非规则性能度量，导致了新的求解方法的研究。

为了描述一个一般的 E/T 模型，令 n 是需要调度的工件数，工件 j_i 用加工时间 p_i 和交货期 d_i 来描述。假设工件是随时可用的，根据调度的结果，工件 j_i 的完成时间是 c_i。令 E_i 和 T_i 分别代表工件 j_i 的提前和拖期时间，并定义为

$$E_i = \max\ \{0, d_i - c_i\} = (d_i - c_i)^+$$

$$T_i = \max\ \{0, c_i - d_i\} = (c_i - d_i)^+$$

与每个工件相对应的是单位提前惩罚 $\alpha_i > 0$，单位拖期惩罚 $\beta_i > 0$。对于一个给定的调度 $\sigma\in\Pi$，令 $f(\sigma)$是对应的目标函数。基本的 E/T 调度问题可以写成：

$$\min_{\sigma\in\Pi} \quad f(\sigma)=\sum_{i=1}^{n}[\alpha_i(d_i-c_i)^+ + \beta_i(c_i-d_i)^-] \tag{7.15}$$

$$=\sum_{i=1}^{n}(\alpha_i E_i+\beta_i T_i) \tag{7.16}$$

然而，可以用不同的方法来度量惩罚。一些 E/T 文献中的大多数方法都是基于对交货期和惩罚费用作出的一般性的假设。在一些 E/T 问题的描述中，交货期是给定的，而另一些却要求交货期和工件顺序同时优化。最简单的模型是考虑所有工件有共同的交货期，更一般的模型是允许有不同的交货期。类似地，一些模型规定相同惩罚，而另一些模型却允许不同工件有不同惩罚，或者不同的 E/T 惩罚，即使最简单模型的优化问题也是一个 NP 难题[73]。

1. 绝对偏差问题

E/T 问题中的一个重要特例是工件完工时间与公共交货期的绝对偏差和的最小化问题。特别地，目标函数可以写成：

$$\min_{\sigma\in\Pi} f(\sigma)=\sum_{i=1}^{n}|c_i-d| \tag{7.17}$$

其中 $d_i=d$。模型中所有工件有相同的 E/T 惩罚系数，希望构造一个调度使得交货期 d 处于中间工件。如果交货期太紧，由于没有工件在时间 0 前开始加工的限制，因而不可能在交货期前放置足够的工件，因此对于给定的问题，公共交货期太紧，就会成为限制问题，否则就是非限制问题。非限制问题存在最优解，且最优解满足如下性质。

（1）调度中没有可插入的空闲时间；

（2）最优调度是 V 型调度；

（3）一个工件恰好在交货期完工；

（4）在最优调度中，第 k 个工件在时间 d 完工，其中 k 是大于或等于 $n/2$ 的最小整数。

该问题的分析可归功于 Kanet[240]，Sundararaghavan 和 Ahmed[390]，Hall[200]，Bagchi，Chang 和 Sullivan[17]，Emmons[119]的工作。

Bagchi，Chang 和 Sullivan 考虑了不同系数的 E/T 惩罚问题：

$$\min_{\sigma\in\Pi} f(\sigma)=\sum_{i=1}^{n}(\alpha E_i+\beta T_i) \tag{7.18}$$

该问题也有非限制和限制之分，在非限制情况下，最优解与绝对偏差问题的最优解有相同的性质。

2. 加权绝对偏差问题

Hall 等[202]研究的加权 E/T 问题定义为：n 个工件及其整数权重 w_i 已知，$i=1,2,\cdots,n$；公共交货期 d 是非限制值。问题的目标是最小化工件的 E/T 加权和，即

$$\min_{\sigma\in\Pi} f(\sigma)=\sum_{i=1}^{n}w_i|c_i-d| \tag{7.19}$$

模型的基本思想是不同的工件应该得到系数不同的惩罚。从一般意义上来说，加权绝

对偏差问题是 NP 完全问题。

令提前工件集是 $E = \{j_i | c_i \leqslant d\}$，拖期工件集 $T = \{j_i | c_i > d\}$。问题有多个最优性条件：

(1) 存在一个最优解其中某些工件恰在交货期 d 完工。

(2) 最优调度是 V 型调度，即 E 中的工件按 w_i/p_i 非减排列，T 中的工件按 w_i/p_i 非增排列。

(3) 给定一个最优调度 σ^*，有

$$\sum_{i \in E} w_i \geqslant \sum_{i \in T} w_i$$

(4) 给定一个最优调度 σ^*，有

$$\sum_{i \in E} p_i \geqslant \sum_{i \in T} p_i$$

3. 平方偏差问题

在某些情况下，不希望出现交货期的太大偏差，这时用与公共交货期的平方偏差作为性能度量可能更为合适：

$$\min_{\sigma \in \Pi} f(\sigma) = \sum_{i=1}^{n} (c_i - d)^2 = \sum_{i=1}^{n} (E_i^2 + T_i^2) \tag{7.20}$$

这是总绝对偏差的二次形式。Elion 和 Chowdhury 曾经证明其最优解是 V 型调度[116]。

Bagchi，Chang 和 Sullivan 还研究了不同 E/T 惩罚的情况[17]，即

$$\min_{\sigma \in \Pi} f(\sigma) = \sum_{i=1}^{n} (\alpha E_i^2 + \beta T_i^2) \tag{7.21}$$

7.1.3 并行机器调度问题

多机调度问题是研究机器对于一组工件的加工顺序，以确保所有工件在合理的总加工时间内加工完成。主要涉及以下三个问题：

(1) 哪台机器分配给哪些工件？

(2) 如何确定工件适当的加工顺序？

(3) 如何评价调度的合理性？

换句话说，多机调度理论主要关心的是如何提供一个机器与工件的完美匹配，或者是近似完美的匹配，然后确定每台机器上工件的加工顺序以达到预先规定的一些目标。

并行机器调度问题可以视为一类由多机调度问题松弛而得到的问题。在并行机器系统中，所有机器都相同并且每个工件可以在任何一个空闲机器上加工；每个加工完的工件释放一台机器并且离开系统。

一个工件可从以下方面来描述：

(1) 工件加工时间：单位加工时间或者各种加工时间；

最优性准则可以分为三个不同组[37]：

(1) 基于完成时间的度量；

(2) 基于交货期的度量；

(3) 基于闲置惩罚的度量。

并行机器调度问题一般作如下假设[144]：

(1) 每个工件只有一道工序；

(2) 一个工件同时只能在一台机器上加工；

(3) 任何机器可以加工任意工件；

(4) 一台机器同时只能加工一个工件；

(5) 机器在调度期间内不会中断，且始终可用；

(6) 机器加工时间与调度无关；

(7) 机器装设时间可忽略；

(8) 机器之间的运输时间可忽略；

(9) 允许加工中的库存存在，其费用可以忽略；

(10) 工件数可预先指定；

(11) 机器数可预先指定；

(12) 对于所有的 i 和 k，机器 m_k 上加工工件 j_i 的时间是预先给定的；

(13) 对所有的 i，工件 j_i 的准备时间已知。

1. 最小流程时间问题

考虑如下的多机调度问题，即考虑同时可用且相互独立的非优先工件的集合 $\{j_1, j_2, \cdots, j_n\}$ 在完全一致、互不关联的并行机器集合 $\{m_1, m_2, \cdots, m_m\}$ $(m < n)$ 上加工的调度问题。对于一个给定的调度 $\sigma \in \Pi$，令 c_j 是工件 j 在调度 σ 下的完工时间，$j=1,2,\cdots,n$；$f(\sigma)$ 表示对应的目标函数值。最小流程时间问题定义为

$$\min_{\sigma\in\Pi} f(\sigma) = \max \{c_j;\ j = 1, \cdots, n\} \tag{7.22}$$

Garey 和 Johnson 已经证明该问题在机器数不定的情况下从严格意义上来说是 NP 难题[147]。当机器数预先给出时可以在伪多项式时间内求解，因此只在一般的意义上是 NP 难题。

2. 最小化加权流程时间问题

令 w_i 和 r_i 分别表示工件 j_i 的权重和准备时间。问题是寻找一个最优的无预先指定的工件调度，使得加权流程时间最小：

$$\min_{\sigma\in\Pi} f(\sigma) = \sum_{j=1}^{n} w_j(c_j - r_j) \tag{7.23}$$

Bruno 等已经证明即使是双机系统在不同工件具有不同的权重情况下也是一个 NP 难题[49]。

3. 最小化最大加权绝对延迟时间问题

给定一个非限制的公共交货期 d，

$$d \geqslant \sum_{j=1}^{n} p_j$$

问题是寻找一个最优的无优先工件的调度，使得最大加权绝对延迟时间最小，即

$$\min_{\sigma \in \Pi} f(\sigma) = \max \{ w_j \mid c_j - d \mid ; j = 1, \cdots, n \} \tag{7.24}$$

Li 和 Cheng 的研究表明，即使对于一个单机系统，该问题也是一个 NP 难题，并且提出了一个启发式方法来求解这一问题[271]。

4. 最小化加权绝对延迟时间和问题

该问题是寻找一个最优的工件调度，使得加权绝对延迟时间和最小，即

$$\min_{\sigma \in \Pi} f(\sigma) = \sum_{j=1}^{n} w_j \mid c_j - d \mid \tag{7.25}$$

最小和问题试图极小化工件完工时间关于交货期的加权绝对偏差和，而最小最大问题是最小化工件完工时间关于交货期的最大加权绝对偏差。

7.2 Cleveland-Smith 方法

Cleveland 和 Smith 首先研究了用遗传算法求解单机调度的问题[76]。事实上他们考虑把遗传算法作为一种下达工件给制造资源(设备)的调度方法，称为部分调度问题。部分调度问题既是一个调度问题，又是一个定时问题。在前一种情况下，问题只限于确定工件下达的顺序，而后一种情况，问题定义为确定每个工件的绝对下达时间。

工件调度问题是寻求下达工件给部分调度的顺序使得加工这些工件的总体费用尽可能少。总体费用常常是库存费用、机器装设费用、提前拖期费用的组合。

7.2.1 遗传算子

这个问题与旅行商问题(TSP)有很多共同之处，可以借助很多求解 TSP 的基于 GA 的方法来求解此问题。TSP 和工件调度的主要区别是工件在调度中的绝对位置和其相对

(4) 从 p_2 中再次选择一块，假定为(3 5 8)，剪去一部分，留下(3 5)，放在第二个位置以避免第一个位置上的障碍。注意，5 被浪费掉，因为它应占有的位置已被填充。

(5) 从 p_1 中再次选择一块，假定为(5 10)，剪去一部分留下 (5)，放到第九个位置。

(6) 从 p_2 中再次选择一块，假定为 (9 2 1)，剪去一部分留下 (2)，放到最后一个位置。

(7) 这时所有的块要么被使用，要么被浪费掉。算子通过随机选择来放置其它剩余的部分。不再使用变异算子。

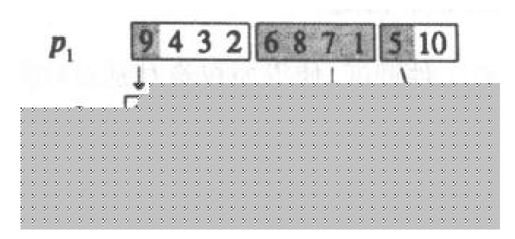


图 7.1 交叉算子：子巡回块

7.2.2 选择

Cleveland 和 Smith 采用 Baker[20]提出的随机通用采样技术作为选择机制。这种方法只需旋转一次转轮，转轮用标准转轮定标，转轮上的刻度数等于种群数。

令 $eval(v_i)$，$i = 1,2,\cdots,pop_size$ 表示染色体 v_i 的适值，期望值 e_i 为

$$e_i = pop_size \times p_i = \frac{pop_size \times eval(v_i)}{\sum_{i=1}^{pop_size} eval(v_i)}; \quad i = 1,2,\cdots,pop_size$$

随机通用采样的步骤可描述为

```
随机通用采样过程
begin
 sum←0;
  ptr←rand();
  for i←1 to pop_size do
  sum←sum+e_i;
    while sum > ptr do
    选择个体 i;
      ptr←ptr+1;
    end
  end
end
```

其中 $rand()$返回一个[0，1)上均匀分布的随机实数值。

7.3 Gupta-Gupta-Kumar 方法

M. Gupta，Y. Gupta 和 A. Kumar 曾用遗传算法求解一个以最小化流程时间方差为

目标的 n 个工件单机调度问题[196]。在他们的研究中，采用了顺序表达法，因为 n 个工件的顺序是单机调度问题的一种自然表达。遗传算法的实现中采用了 PMX 算子、交换变异、线性标定适值函数、简单多样化策略及聚集与精选的组合选择机制。

7.3.1 评价函数

因为 CTV 和 WCTV 问题都是最小化问题，用以下变换把原来的目标函数映射成一个适值函数：

$$f(v_i) = \begin{cases} c_{max} - g(v_i), & g(v_i) < c_{max} \\ 0, & \text{其他} \end{cases} \tag{7.26}$$

其中 $g(v_i)$和 $f(v_i)$分别表示染色体 v_i 的原来目标函数和适值函数。

采用 Goldberg[171]提出的线性标定法，用平均适值和最好适值来区分染色体。

$$f' = af + b \tag{7.27}$$

其中 f 是原适值，f' 是标定的适值，a 和 b 是常数。

7.3.2 替代策略

用聚集(crowding)和精选(elitist)的组合来作为替代策略[272]。通过遗传运算，在所产生后代池中构造新的种群。如果所有的后代都比前一代的每个染色体优越，那么在新的种群中，所有的后代代替原有的染色体。如果其中一部分相当好，那么它们替代前一代种群中相同数量的性能最差的染色体。对于剩余的后代，用预设置的概率值从前一代种群中选择后代。替代策略确保新种群继承前一代种群中最好的染色体。

7.3.3 收敛性策略

递增顺序将种群分类；

4）检查 oldpop 的多样性是否达到可接受水平；如果不可接受，执行步骤 4。

步骤 2：重组

1）实行 PMX 交叉和交换变异；

2）计算每个后代的目标函数值和适值；

3）按照目标函数值的递增顺序把选择池分类。

步骤 3：替代

把分类过的老种群和选择池中的染色体按适值作比较，用替代策略产生一个新种群：

1）如果所有的后代比前一代种群中的每个染色体的性能都要优越，那么在新的种群中用所有的后代替代现有的染色体；

2）如果仅有部分比前一代种群的染色体优越，那么用这一部分替代前一代种群中最差的相同数量的染色体；

3）对于其余的后代，用预先设置的概率选择。

步骤 4：多样化

应用变异过程来保持种群的多样化：

1）用式(7.28)和(7.29)计算当前种群的多样性参数 H；

2）与给定的可接受的水平比较，如果低于可接收水平，重复执行变异过程，直到种群的多样性达到可接收水平。

步骤 5：如果 $gen < max_gen$，令 $gen \leftarrow gen+1$ 返回步骤 2，否则停止。

该算法对规模在 10 个工件到 20 个工件的不同问题进行了检验，基本参数设置如表 7.1 所示。

表 7.1 GA 的基本参数设置

参数	范围
pop_size	(75, 100)
max_gen	(100, 400)
p_c	(0.6, 0.9)
p_m	(0.05, 0.15)

7.4 Lee-Kim 方法

Lee 和 Kim 曾经提出了一个并行遗传算法用于求解单机的工件调度问题[266]。调度的目标是最小化公共交货期下的加权 E/T 惩罚和。采用二进制表达法把工件调度编码为染色体，通过保持种群在不同的子组中使遗传算法并行化。

一般加权 E/T 问题(GWET)定义为

$$\min_{\sigma \in \Pi} f(\sigma) = \sum_{i=1}^{n} \{\alpha_i (d - c_i)^+ + \beta_i (c_i - d)^+\}$$

对权重 α_i 和 β_i 没有限制。众所周知这个问题属于 NP 难题[201,202,265]。

7.4.1 表达法

GWET 问题的最优调度是以交货期为中心的 V 型调度[18,22]，也就是说，在交货期或交货期前完工的工件按照 p_i/α_i 非增的顺序加工，在交货期或交货期以后开始加工的工件按照 p_i/β_i 的非减顺序加工。因为最优调度是 V 型调度，Lee 和 Kim 提出了一种二进制表达法以确保所有的染色体是 V 型调度。

对于 n 个工件问题，染色体包含 n 个二进制位，每一位标明相应工件属于拖期工件集合 T 或非拖期工件集合 E。例如，如果第 i 位是 0，对应的工件 i 属于集合 E；否则属于集合 T。该表达法通过在集合 E 中按照 p_i/α_i 的非增顺序排列工件，在集合 T 中按照 p_i/β_i 的非减顺序排列工件来构造调度。下面考虑表 7.2 给出的 9 个工件的简单实例。

表 7.2　9 个工件的实例

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9
p_i	2	2	4	5	1	4	3	2	1
α_i	1	2	6	4	4	8	10	5	9
β_i	8	8	8	2	9	2	2	1	2
$p_i\beta_i/\alpha_i$	16	8	5.3	2.5	2.25	1	0.6	0.4	0.2
p_i/α_i	2	1	0.6	1.25	0.25	0.5	0.3	0.4	0.1
p_i/β_i	0.25	0.25	0.5	2.5	0.11	2	1.5	2	0.5

假设给定染色体为

$$[1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0]$$

因为第 1,4,7,8 位是 1，因此拖期工件集合为 $T=\{j_1,j_4,j_7,j_8\}$，于是非拖期工件集合 $E=\{j_2,j_3,j_5,j_6,j_9\}$。通过在集合 E 中按照 p_i/α_i 的非增顺序调度工件，在集合 T 中按照 p_i/β_i 的非减顺序调度工件形成了一个 V 型调度：

$$[j_2\ j_3\ j_6\ j_5\ j_9\ j_1\ j_7\ j_8\ j_4]$$

这种编码表达法的两个主要组成部分是

(1) 利用二进制串，工件要么属于集合 E，要么属于集合 T；

(2) 通过集合 E 中 p_i/α_i 的非增顺序和集合 T 中 p_i/β_i 的非减顺序来排列工件，构造一个 V 型调度。

7.4.2 并行子种群

并行遗传算法包含了一组子种群，每个子种群利用遗传算子独立地产生后代。每个子种群用其调度的第一个工件来表征，也就是说每一组串有和第一个加工的工件相同的工件。

问题是如何确定子种群数 m，由 Lee 和 Kim 提出的随机 V 型调度产生器，试图在时间段$[0,d]$中放置一些 p_i/α_i 非增顺序的工件，把其它的工件按照 p_i/β_i 非减的顺序放置在时间段 $[d,\ \sum_{i=1}^{n} p_i]$ 中。一个调度是由不能形成 V 型调度的工作开始的，因为在这种情

况下集合 T 中工件的总加工时间将超过 $\sum_{i=1}^{n} p_i - d$。Lee 和 Kim 提出了一个引理来识别关键工件 k，然后 按如下过程形成子组：首先按照 p_1/α_1，p_2/α_2，…，p_n/α_n 的顺序排列工件，子种群 1 中的串从具有最大 p_i/α_i 值的工件开始；在子种群 2 中，每一串的第一个工件有第二大的 p_i/α_i 值；在子种群 3 中将有第三或第四大的 p_i/α_i 值的工件排在首位；在子种群 m 中，则将有第($2^{m-2}+1$)大，第($2^{m-2}+2$)大，……，第 2^{m-1}大的 p_i/α_i 值的工件排在首位，重复执行这一过程直到遇到关键工件 k。

初始化过程包括以下五个主要步骤：

初始化过程

步骤 1：对于每个子组，首先分配 0 给起始工件（即把它放到非拖期工件集合 E 中）。

步骤 2：将所有 p_i/α_i 值大于起始工件对应值的工件置为 1(即放到拖期集合 T 中)。

步骤 3：其他工件随机地分配到集合 E 或集合 T 中，但 E 中工件的总加工时间不超过交货期。

步骤 4：当具有位值为 0 的工件的完工时间首先超过交货期 d 时，其位值改成 1，其他所有剩余的未被分配的工件的位值也都设为 1。

步骤 5：当具有位值 1 的工件的总完工时间超过 $\sum_{i=1}^{n} p_i - d$ 时，所有剩余工件的位值设为 0。

7.4.3 交叉与变异

既然染色体被编码成二进制串，那么就可以应用单点交叉和双点交叉作为交叉算子。然而，通过交叉产生的后代不能保证是 V 型调度，因为集合 E 中工件的总加工时间可能超出或远远少于交货期。为了调整后代是一个 V 型调度，可以采用以下的变异算子：

(1) 如果集合 E 中工件的总加工时间超过交货期，把满足 $\{i \mid c_i - p_i \geq d\}$ 的工件放进集合 T，并令满足 $c_i - p_i \geq d < c_i$ 的工件 i 拖期。

(2) 如果集合 T 中工件的总加工时间超过 $\sum_{i=1}^{n} p_i - d$，把满足 $\{i \mid c_i \geq d\}$ 的工件放进集合 E。

变异则将一些适当的工件移动到交货期前或后，以保持 V 型的调度。

7.4.4 评估与选择

为了评估每个染色体的惩罚，应首先按集合 E 中 p_i/α_i 的非增顺序和集合 T 中 p_i/β_i 的非减顺序排列工件，把染色体解码成一个调度。然后应用线性标定法来确定每个调度的适值：

(1) 用每个子种群中的最大惩罚减去染色体的惩罚得到每个调度的原始适值。

(2) 用线性标定法计算适值，其中线性标定法中最小的适值为 0，平均适值等于平均原始适值。

测试了两个繁殖方法:转轮选择和确定性选择(N 个最好选择)。

7.4.5 并行遗传算法

基于岛屿模型(Island model)的并行遗传算法,具有如下结构:

```
并行遗传算法
begin
初始化
评估
  while 每个子种群未完    do
    繁殖
    交叉
    变异
    评估
    交互
    删除
  end
end
```

交互是在并行遗传算法子种群之间交换信息的一种方法。每个子种群中最好的染色体被转换到其他的子种群,通过对特定组染色体的配对产生改进的调度。

有两类较为流行的交互方法:所有节点连接和邻近节点连接[303]。在所有节点连接中,每个子种群中的最好染色体被转换给每个其它的子种群。而在邻近节点连接中,只有相邻的子种群共享最好的染色体。为了避免由于其它子种群出现特优的个体而导致的提前收敛,Lee 和 Kim 采用了邻近节点连接方式。子种群被连接成一个环,每个子种群送出一份最好的染色体给邻近的组,同时也得到多份最好的染色体。

7.5 Cheng-Gen 方法

Cheng 和 Gen 应用遗传算法,求解了一个以最小化最大加权绝对延迟时间为目标的、并行机器系统的工件调度问题[68,443]。Li 和 Cheng 首先考虑了这个问题[271],即有一组已知加工时间和权重的工件集合,并有多台相同的并行机器和公共交货期。目标是寻找一个最优调度,使得最大加权绝对延迟时间最小,其目标函数是非规则性能度量。

7.5.1 表达法与初始化

众所周知,对于各种类型的多机调度问题主要是解决以下两个本质问题:

(1) 分配工件给机器;

(2) 对于每台机器调度工件。

本节提出一种扩展的顺序表达法,把这两个方面编码成染色体,其中整数代表所有可能的工件排列,* 标志工件分配给机器。现考虑 9 个工件、3 台机器的例子,假定有一个如图 7.2 所示的调度,其染色体表示为

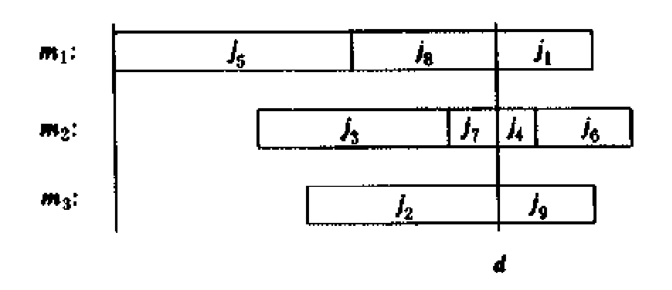


图 7.2　9 个工件、3 台机器问题的一个调度

$$[5\ 8\ 1\ *\ 3\ 7\ 4\ 6\ *\ 2\ 9]$$

一般地，对于 n 个工件、m 台机器问题，一个合法的染色体应包括 n 个工件符号，$m-1$ 个分配符号，总数为 $(n+m-1)$。

随机地产生初始种群。总体步骤为

初始化步骤

```
begin
i ← 0;
  while i ≤ pop_size do
  begin
    随机产生一个工件列表；
    在工件列表中随机插入 m－1 个 *；
    i ← i+1;
  end
end
```

7.5.2　交叉

如上所述，多机调度问题的本质问题是工件和机器的组合和排列，可用交叉和变异算子来调整工件的分配和排列。

多机系统下单台机器的一个调度称为子调度。子调度可以看作是遗传搜索的自然构成块，为了保持后代中这样的构成块像 Holland 描述的那样，给出了一个子调度维持交叉算子。交叉算子产生后代的基本思想是，首先取两个双亲来重构总体分配结构，把一个双亲中的一个子调度扩展成一个后代，然后用另一个双亲中剩余的工件来完成该后代的构造。

交叉过程

步骤 1：从一个双亲中得到 * 位置（或总体分配结构）；

步骤 2：从相同的双亲中得到一个随机选择的子调度；

步骤 3：通过从左到右扫描，从另一个双亲中得到剩余的工件。

假设有两个双亲 p_1 和 p_2，选择 p_1 的最右边子调度扩展成后代 o，交叉运算的执行过程如图 7.3 所示。从图中可以看出，提出的交叉算子能够同时调整工件的分配和顺序。

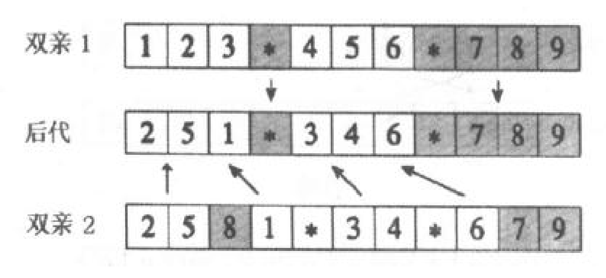


图 7.3　交叉运算图解

7.5.3　变异

采用两种变异方法，一种是交换变异，随机选取两个位置，然后交换基因；另一种是启发式变异，使用贪婪启发式来调整工件的顺序，对于每台机器形成 V 型调度。

1. 交换变异

随机交换的基因可以是工件，也可以是 * 号，工件和 * 号的不同组合导致了四种基本类型的变异：

（1）如果两个基因都是工件，可能出现两种情况。一种情况是两个工件在同一台机器上加工，这时变异改变工件的顺序，如图 7.4(a)所示。

（2）另一种情况是两个工件在不同的机器上加工。这时变异改变染色体的工件顺序和工件对机器的分配，如图 7.4(b)所示。

（3）如果两个基因都是 * 号，变异执行无意义的操作，如图 7.4(c)所示，这种操作是禁止的。

（4）如果一个基因是 *，另一个基因是工件，变异同时改变染色体的工件顺序和工件对机器的分配，如图 7.4(d)的所示。

最后一种类型是改变 * 号位置的唯一遗传操作。没有这种操作，* 的位置在整个进化过程中都不能改变，因此遗传搜索受到初始种群的限制。这一类的变异在遗传搜索中起到关键作用。

变异过程

```
begin
i ← 0;
  while i ≤ pop_size × p_m do
  随机选择一个染色体；
  随机挑出两个基因；
    if 两个基因都是 * then
    随机挑选一个工件；
    end
    交换位置；
    i ← i+1;
  end
end
```

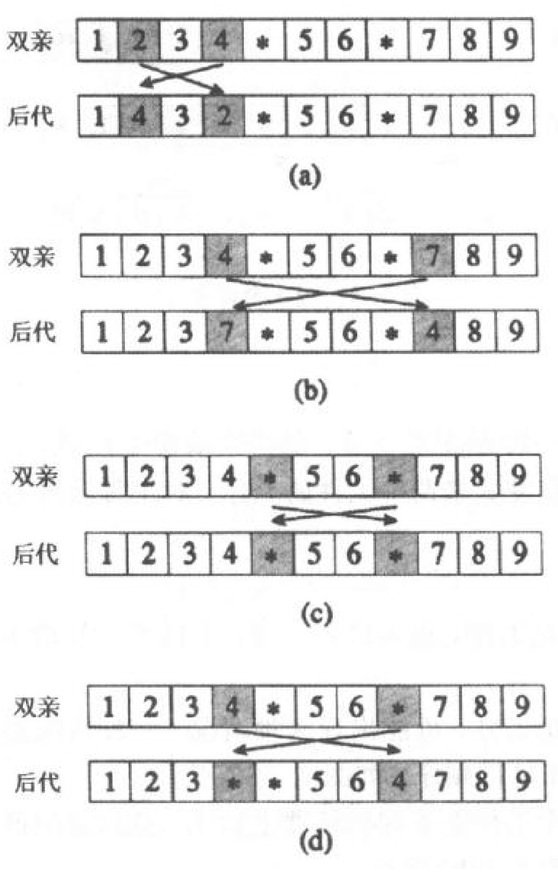


图 7.4　变异算子图解

2. 启发式变异

这种变异采用贪婪策略来调整工件的顺序，从而形成对于每台机器的 V 型子调度。也就是说，公共交货期前的所有工件按照 p_i/w_i 的非增顺序排列，而公共交货期后的工件按照 p_i/w_i 的非减顺序排列。

令 $d-x$ 表示最早工件的起始时间，$d+y$ 表示调度的最后一个工件的完工时间，如图 7.5 所示。启发式变异的步骤如下：

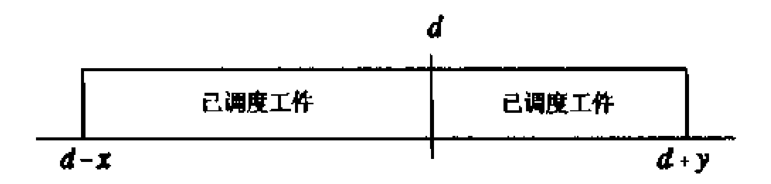


图 7.5　调度工件的起始和完工时间

启发式变异过程

步骤 1：按 w_i/p_i 的递减顺序排列工件。令调度的工件是$(j_1,j_2,\cdots,j_n)$。置 $x\leftarrow 0$，$y\leftarrow 0$。

步骤 2：把工件 j_1 放在交货期 d 前的位置，$x\leftarrow x+p_1$。

步骤 3：**for**　$i\leftarrow 2$　**to**　n

　　　　if $(y+p_i< x)$**then**

```
        把工件 j_i 加在调度工件列表的最后；
          y ← y + p_i；
        else
          把工件 j_i 加在调度工件列表的最前面；
          x ← x + p_i；
        end
      end
```

由表 7.3 给出的 5 个工件的实例，其调度的工件是(j_5,j_3,j_2,j_4,j_1)，用启发式构造的调度如图 7.6 所示。

表 7.3　5 个工件的实例

	j_1	j_2	j_3	j_4	j_5
p_i	2	2	4	5	1
w_i	1	2	6	4	4
w_i/p_i	0.5	1.0	1.5	0.8	2.0

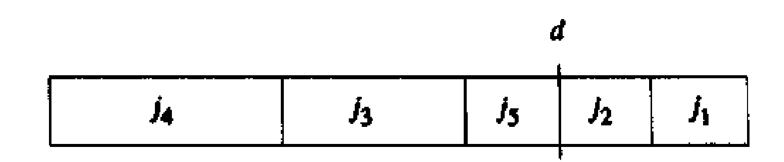


图 7.6　用贪婪启发式构造的 V 型调度

7.5.4　确定最好交货期

直到现在我们考虑的是如何用遗传算子处理工件分配和工件调度的问题，还没有讨论如何确定最好交货期问题。对于在一台机器上给定的一个工件顺序，能够最优地确定公共交货期。令交货期 d 是决策变量，对于任意两个工件 i 和 j，最好交货期由如下公式确定：

$$w_i(d - c_i) = w_j(c_j - d) \tag{7.30}$$

即工件 i 和 j 的交货期为

$$d = \frac{w_i c_i + w_j c_j}{c_i + c_j} \tag{7.31}$$

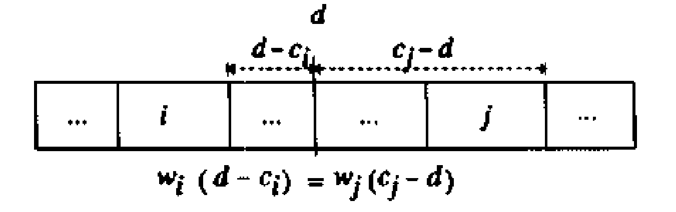


图 7.7　最好公共交货期

依次可以计算所有可能工件对的交货期，给定工件调度的最好交货期可从方程(7.30)表示的具有最大绝对延迟的交货期中选取。

对于多机情况，首先用以上方法计算对于机器 k 的最好交货期 d_k，m 台机器的公共交货期由下式确定：

$$d = \max \{d_k;\ k = 1, 2, \cdots, m\} \tag{7.32}$$

关于公共交货期 d，如果工件在机器 k 上加工，需要推迟每个工件的起始时间

$$\Delta_k = d - d_k \tag{7.33}$$

确定最好公共交货期的步骤可用图 7.8 所示的一个简单实例来说明。

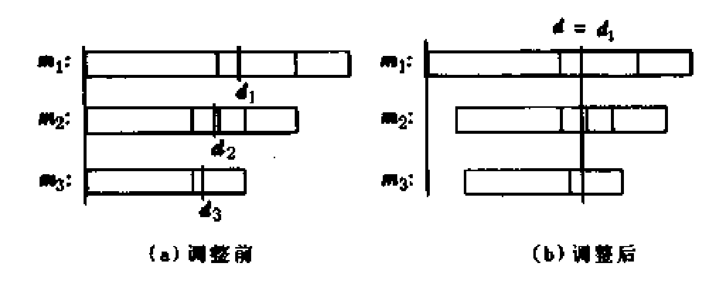


图 7.8 起始时间的调整

7.5.5 评估与选择

每个染色体的适值用其最大加权绝对延迟时间的倒数来计算：

$$eval(v_t) = \frac{1}{f(v_t)}; \qquad t = 1, 2, \cdots, pop_size \tag{7.34}$$

其中，$eval(v_t)$是第 t 个染色体的适值函数；$f(v_t)$是目标函数值[参见等式(7.24)]。用转轮策略和精选策略的组合策略作为基本的选择机制。

7.5.6 计算实例

本节提出的遗传算法对随机产生的 30 个工件、5 台机器的调度问题进行了检验。遗传算法的基本参数设置是 $pop_size = 40$，$p_c = 0.4$，$p_{m_1} = 0.4$，$p_{m_2} = 0.4$，$max_gen = 200$。计算 20 次的结果如图 7.9 所示。

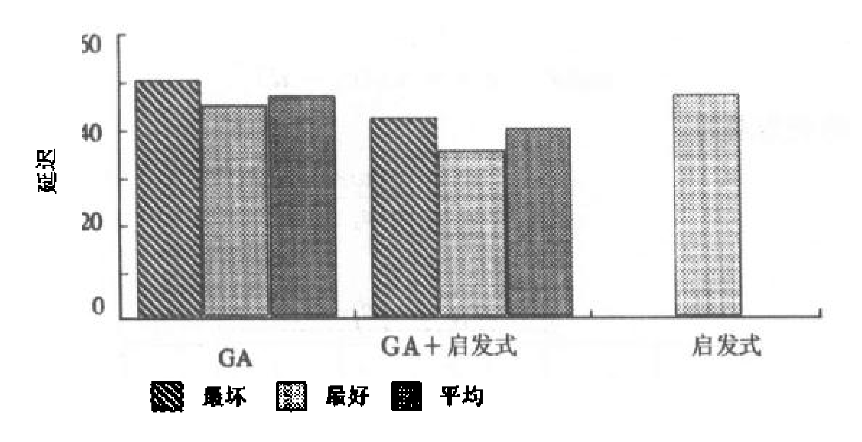


图 7.9 GA 与启发式算法的比较

其中 GA 代表不用变异 2 的方法，GA＋启发式代表包括变异 2 的方法，启发式代表 Li 和 Cheng 的贪婪启发式。结果表明混合式遗传算法最好。

图 7.10 表示混合式遗传算法在 50 次运行下解的分布频率。

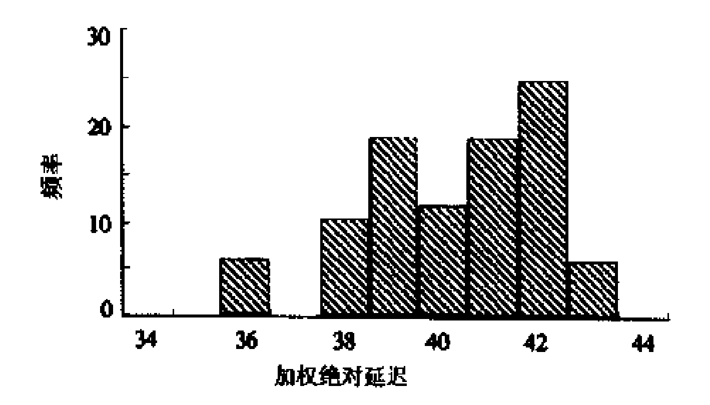


图 7.10 解的分布情况

第八章　运输问题

8.1　引　　言

运输问题最早是由 Hitchcock 在 1941 年提出的[217]。此后，人们对这一问题的研究给予了极大的关注并对各类基本运输问题进行了研究。根据目标的类型，可将问题分为

(1) 线性问题或非线性问题；

(2) 单目标问题或多目标问题。

根据约束的类型，又可将问题分为

(1) 二维(planar)问题或三维(solid)问题；

(2) 平衡问题或非平衡问题。

基本的运输问题是线性单目标二维平衡问题。由于问题约束具有特殊的结构，人们提出了一种有效的优化算法，这种算法是适合于特殊结构的单纯形法的变形[26,419]。

Vignaux 和 Michalewicz 首先讨论了用遗传算法解决线性运输问题(LTP, Linear Transportation Problem)[410]。当然，他们的目的不是为了比较遗传算法和传统优化算法，而是用它作为一个带约束优化问题的例子，研究如何用遗传算法处理约束。结果表明了遗传算法的威力，它允许采用任何符合问题的数据结构和任何有意义的遗传算子集合。Michalewicz，Vignaux 和 Hobbs 针对标准运输算法不能解决的平衡非线性运输问题，也开发了一种遗传算法[294]，称作 GENETIC-2，尽管 GENETIC-2 是为运输问题特别研究的，但它的一个重要特性是可以处理任何类型的费用函数，对它修改后有可能解决许多基于矩阵的约束优化问题。

Yang 和 Gen 将 Michalewicz 的工作推广到双准则线性运输问题(BLTP，Bicriteria Linear Transportation Problem)[426]和双准则三维运输问题(BSTP，Bicriteria Solid Transportation Problem)[155]。人们还提出了一些用多目标线性规划来解决运输问题的方法[8,77,355]。多目标问题的通常方法是在决策空间或目标空间产生非劣端点。根据这一方法，他们试图用遗传算法来确定这样的端点集合。他们将目标空间方法的基本思想嵌入评估阶段，以便促使遗传搜索在目标空间中向非劣点方向进行，并阐明用传统优化技术加强遗传算法是解决这样的复杂多目标优化问题的有效途径。

8.2　线性运输问题

8.2.1　LTP 的描述

线性运输问题 LTP 是指从不同的供给起点或来源向不同的终点运送同一种物品(commodity)，每个终点需要特定数量的物品。问题是如何分配在每个起点的供给，以便在满足每个终点需求的条件下，优化某个目标。常用的目标函数是最小全部运输费用、最小全部加权距离或最大全部利润[51]。

给定 m 个起点和 n 个终点，运输问题可描述为线性规划模型：

$$\min \quad z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{8.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = a_i; \quad i = 1,2,\cdots,m \tag{8.2}$$

$$\sum_{i=1}^{m} x_{ij} = b_j; \quad j = 1,2,\cdots,n \tag{8.3}$$

$$x_{ij} \geqslant 0; \quad \forall\, i,j \tag{8.4}$$

其中，x_{ij}是从起点 i 到终点 j 的运输量；c_{ij}是从起点 i 向终点 j 运送单位量所需的费用；a_i是起点 i 可供给量；b_j 是终点 j 的需求量。约束(8.2)是供给约束；约束(8.3)是需求约束。

1. 运输问题的可行性

上述描述假设全部供给和全部需求是相等的，即

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \tag{8.5}$$

在平衡假设条件下，运输问题通常有可行解。例如容易发现：

$$x_{ij} = \frac{a_i b_j}{\sum_i a_i}; \quad i = 1,2,\cdots,m; \quad j = 1,2,\cdots,n$$

就是一个可行解。需注意的是对于每个可行解，每个运输量 x_{ij}要受如下限制：

$$0 \leqslant x_{ij} \leqslant \min\,\{a_i,\, b_j\}$$

我们知道具有可行解的受限线性规划存在最优解[26]。

问题通常用如图 8.1 所示的运输表表示，其中行代表起点，列代表终点，i 行 j 列格中的内容代表决策变量 x_{ij}，相应的费用系数 c_{ij}放在(i,j)格的右上角处。

终点

起点

从＼到	1	2	…	n	供应
1	x_{11} c_{11}	x_{12} c_{12}		x_{1n} c_{1n}	a_1
2	x_{21} c_{21}	x_{22} c_{22}		x_{2n} c_{2n}	a_2
…					…
m	x_{m1} c_{m1}	x_{m2} c_{m2}		x_{mn} c_{mn}	a_m
需求	b_1	b_2	…	b_n	

图 8.1 遗传运输表

运输问题也可用图来描述(如图 8.2 所示)。这样的图由起始节点 $O_i, i=1,2,\cdots,m$ 和终止节点 $D_j, j=1,2,\cdots,n$ 及连接他们的边构成。因为节点可分为两个集合，一个集合中的所有边直接与另一个集合相连，所以它是完全二分图(complete bipartite graph)。说它是“完全的”是指所有这样的边都存在。

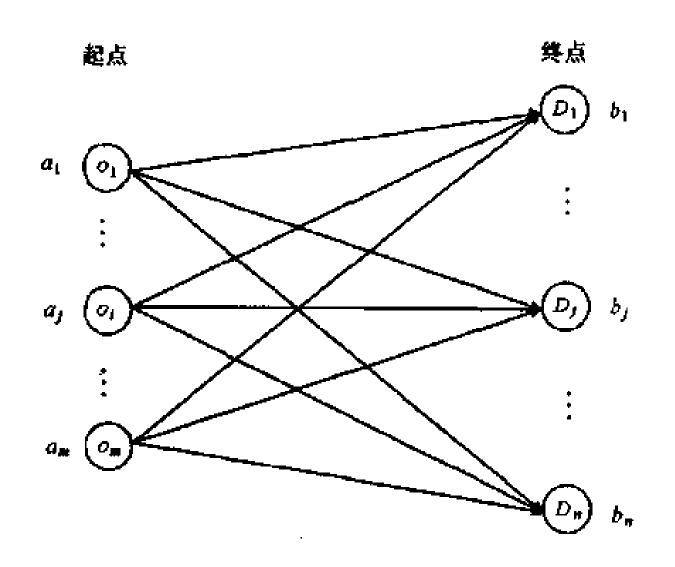


图 8.2 运输问题的网络模型

Vignaux 和 Michalewicz 提出了解决运输问题的遗传算法的两种实现:GENETIC-1 和 GENETIC-2[410]。在 GENETIC-2 中,他们用矩阵描述染色体并在遗传算法中引入问题的特定知识来处理约束。

8.2.2 表达方式

矩阵或许是运输问题的解的最适当的表达方式。运输问题的分配矩阵可描述如下:

$$X_p = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

其中 X_p 代表第 p 个染色体,x_{ij}是相应的决策变量。

基于非负条件(8.4)和平衡条件(8.5),可用下述初始化过程产生满足所有约束的初始种群。

初始化过程

begin

$\pi \leftarrow \{1, 2, \cdots, mn\}$;

  repeat

  从集合 π 中任选一个数 k;

    计算相应的行和列;

      $i \leftarrow \lfloor (k-1)/n + 1 \rfloor$;

      $j \leftarrow (k-1) \bmod n + 1$;

    将可用量赋给 x_{ij};

      $x_{ij} \leftarrow \min \{a_i, b_j\}$;

    修改数据:

      $a_i \leftarrow a_i - x_{ij}$;

      $b_j \leftarrow b_j - x_{ij}$;

$\pi \leftarrow \pi \setminus \{k\}$;

until π 为空

end

这一过程的基本思想是

(1) 从分配矩阵中任选一个决策变量 x_{ij};

(2) 赋给 x_{ij}尽可能多的可用量;

(3) 修改需求和供应数据以保证平衡条件。

用目标函数(8.1)来评估每个染色体,用 $eval(X_p)$表示染色体 X_p 的适值函数,则有

$$eval(X_p) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij} \tag{8.6}$$

8.2.3 遗传运算

1. 交叉

假设矩阵 $X_1 = (x_{ij}^1)$和 $X_2 = (x_{ij}^2)$选作交叉运算的双亲,交叉主要由以下三步完成:

步骤 1: 建立两个临时矩阵 $D = (d_{ij})$和 $R = (r_{ij})$:

$$d_{ij} = \lfloor (x_{ij}^1 + x_{ij}^2) / 2 \rfloor \tag{8.7}$$

$$r_{ij} = (x_{ij}^1 + x_{ij}^2) \bmod 2 \tag{8.8}$$

矩阵 D 存放两个双亲取整的均值,矩阵 R 记录是否需要取整。两个矩阵的关系由下列等式给出:

$$a_i - \sum_{j=1}^{n} d_{ij} = \frac{1}{2}\sum_{j=1}^{n} r_{ij}; \quad i = 1,2,\cdots,m \tag{8.9}$$

$$b_j - \sum_{i=1}^{m} d_{ij} = \frac{1}{2}\sum_{i=1}^{m} r_{ij}; \quad j = 1,2,\cdots,n \tag{8.10}$$

这些等式描述了两个有趣的特性:

(1) 每行每列中 1 的数目是偶数,即行与列的边际和 $\sum_{j=1}^{n} r_{ij}$ 与 $\sum_{i=1}^{m} r_{ij}$ 都是偶整数。

(2) 矩阵 R 行边际和的值等于矩阵 D 行边际和与相应的供应之差 $a_i - \sum_{j=1}^{n} d_{ij}$ 的两倍,矩阵 R 列边际和等于矩阵 D 列边际和与相应需求之差 $b_j - \sum_{i=1}^{m} d_{ij}$ 的两倍。

用式(8.7)和(8.8)中的定义很容易证明这些特性。下面我们给出等式(8.9)的证明:

$$\begin{aligned}
\sum_{j=1}^{n} d_{ij} &= \sum_{j=1}^{n} \left\lfloor \frac{1}{2}(x_{ij}^1 + x_{ij}^2) \right\rfloor \\
&= \sum_{j=1}^{n} \frac{1}{2}\left((x_{ij}^1 + x_{ij}^2) - (x_{ij}^1 + x_{ij}^2) \bmod 2 \right) \\
&= \frac{1}{2}\sum_{j=1}^{n}(x_{ij}^1 + x_{ij}^2) - \frac{1}{2}\sum_{j=1}^{n}\left((x_{ij}^1 + x_{ij}^2) \bmod 2 \right) \\
&= a_i - \frac{1}{2}\sum_{j=1}^{n} r_{ij}
\end{aligned}$$

步骤 2:将矩阵 R 分成两个矩阵 $R^1=(r_{ij}^1)$和 $R^2=(r_{ij}^2)$,它们满足:

$$R = R^1 + R^2 \tag{8.11}$$

$$\sum_{j=1}^{n} r_{ij}^1 = \sum_{j=1}^{n} r_{ij}^2 = \frac{1}{2}\sum_{j=1}^{n} r_{ij}; \quad i = 1,2, \cdots, m \tag{8.12}$$

$$\sum_{i=1}^{m} r_{ij}^1 = \sum_{i=1}^{m} r_{ij}^2 = \frac{1}{2}\sum_{i=1}^{m} r_{ij}; \quad j = 1,2, \cdots, n \tag{8.13}$$

容易看出,在满足上述条件的情况下,将 R 分成 R^1 和 R^2 有许多可能的情况。

步骤 3:然后产生两个如下后代 X_1' 和 X'_2:

$$X_1' = D + R^1 \tag{8.14}$$

$$X_2' = D + R^2 \tag{8.15}$$

让我们看一个简单的例子。

例 8.1 假设问题有 4 个起点和 5 个终点,并满足如下约束:

$$a_1 = 8, a_2 = 4, a_3 = 12, a_4 = 6$$

$$b_1 = 3, b_2 = 5, b_3 = 10, b_4 = 7, b_5 = 5$$

下面的矩阵选为交叉的双亲:

双亲 X_1

1	0	0	7	0
0	4	0	0	0
2	1	4	0	5
0	0	6	0	0

双亲 X_2

0	0	5	0	3
0	4	0	0	0
0	0	5	7	0
3	1	0	0	2

临时矩阵 D 和 R:

矩阵 D

0	0	2	3	1
0	4	0	0	0
1	0	4	3	2
1	0	3	0	1

矩阵 R

1	0	1	1	1
0	0	0	0	0
0	1	1	1	1
1	1	0	0	0

而后将 R 分成 R^1 和 R^2:

矩阵 R^1

0	0	1	0	1
0	0	0	0	0
0	1	0	1	0
1	0	0	0	0

矩阵 R^2

1	0	0	1	0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0

最后,两个后代为 X_1' 和 X_2':

后代 X_1'

0	0	3	3	2
0	4	0	0	0
1	1	4	4	2
2	0	3	0	1

后代 X_2'

1	0	2	4	1
0	4	0	0	0
1	0	5	3	3
1	1	3	0	1

2. 变异

变异由以下三步完成：

步骤 1：从双亲矩阵中建立一个子矩阵。任选行$\{i_1,\cdots,i_p\}$和列$\{j_1,\cdots,j_q\}$建立$(p*q)$子矩阵 $Y=(y_{ij})$，其中$\{i_1,\cdots,i_p\}$是$\{1,2,\cdots,m\}$的真子集，并且 $2\leqslant p\leqslant m$；$\{j_1,\cdots,j_q\}$是$\{1,2,\cdots,n\}$的真子集，并且 $2\leqslant q\leqslant n$。y_{ij}从双亲矩阵中 i 行 j 列交叉点的相应元素取值。

步骤 2：重新分配子矩阵的物品(commodity)。子矩阵中物品 a_i^y 和需求 b_j^y 的可用量确定如下：

$$a_i^y=\sum_{j\in\{j_1,\cdots,j_q\}}y_{ij};\quad i=i_1,i_2,\cdots,i_p$$

$$b_j^y=\sum_{i\in\{i_1,\cdots,i_p\}}y_{ij};\quad j=j_1,j_2,\cdots,j_q$$

我们可以用初始化过程对子矩阵赋新值以满足 a_i^y 和 b_j^y 的所有约束。

步骤 3：用重新分配的子矩阵 Y 中的新元素替换双亲矩阵中的适当元素。

例 8.2 仍然以上述问题为例。假设下面的矩阵 X 选为变异的双亲：

双亲 X

0	0	5	0	3
0	4	0	0	0
0	0	5	7	0
3	1	0	0	2

任选两行$\{2,4\}$三列$\{2,3,5\}$。相应的子矩阵和重分配子矩阵为

子矩阵 Y

4	0	0
1	0	2

重分配子矩阵

2	0	2
3	0	0

则变异后的后代为

后代

0	0	5	0	3
0	2	0	0	2
0	0	5	7	0
3	3	0	0	0

8.3 双准则线性运输问题

8.3.1 BLTP 的描述

极小化全部费用的单目标运输问题在文献中多有报导[198,419]。然而,在现实情况下,复杂的社会和经济环境需要采用除费用以外的其他准则。它们包括:物品的平均交货时间,运输的可靠性,用户可接近性(accessibility),产品损失(deterioration)等,详细内容见 Current 和 Min 的文章[88,89]。这些运输问题实际上是多目标问题,因而,可表述为多目标线性规划问题。

由于可行域在目标空间可用二维描述,双准则线性运输问题(BLTP)是多目标运输问题的一个特例。考虑下述两个目标:最小化全部费用和全部损失。假设有 m 个起点和 n 个终点,BLTP 可用如下公式描述:

$$\min \quad z^1 = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^1 x_{ij} \tag{8.16}$$

$$\min \quad z^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^2 x_{ij} \tag{8.17}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = a_i; \quad i = 1,2,\cdots,m \tag{8.18}$$

$$\sum_{i=1}^{m} x_{ij} = b_j; \quad j = 1,2,\cdots,n \tag{8.19}$$

$$x_{ij} \geqslant 0; \quad \forall\, i,j \tag{8.20}$$

其中 x_{ij}是从起点 i 到终点 j 的运送量;c_{ij}^1是从起点 i 到终点 j 运送单位量的费用;c_{ij}^2是从起点 i 到终点 j 运送单位量的损失;a_i 是在起点 i 的可用量;b_j 是在终点 j 的需要量。约束(8.18)是供应约束,约束(8.19)是需求约束。

8.3.2 评估

Yang 和 Gen 进一步扩展了 Michalewicz 关于 BLTP 的工作[426]。BLPT 的常规方法是在决策空间或目标空间产生非全优的端点[8,79]。根据这一思路,他们用遗传算法寻找非劣点集合。如我们所知,遗传算法在寻找可行解方面是非常有效的。因而,下面的问题就是如何增加选择压力以迫使遗传算法探索非劣集中的有效端点。Yang 和 Gen 采用自适应移动线技术建立一种求加权和的方法,这种方法可以迫使遗传搜索去探索第二章讨论的区间规划问题在目标空间中 Pareto 解的集合。

令 E 代表迄今为止找到的非劣解集合。E 中的两个特殊点引起我们的兴趣:一个点包括 $z_{\min}^1$而另一个点包括 $z_{\min}^2$。这两个点分别表示为$(z_{\min}^1, z_{\max}^2)$和$(z_{\max}^1, z_{\min}^2)$,其中

$$z_{\min}^1 = \min\ \{z^1(\mathbf{x}) \mid \mathbf{x} \in E\}$$

$$z_{\max}^1 = \max\ \{z^1(\mathbf{x}) \mid \mathbf{x} \in E\}$$

$$z_{\min}^2 = \min\ \{z^2(\mathbf{x}) \mid \mathbf{x} \in E\}$$

$$z_{\max}^2 = \max\ \{z^2(\mathbf{x}) \mid \mathbf{x} \in E\}$$

基于这两个特殊点我们可以建立如下新的目标函数：

$$z = \alpha z^1 + \beta z^2 \tag{8.21}$$

其中

$$\alpha = |z^2_{\max} - z^2_{\min}|$$

$$\beta = |z^1_{\max} - z^1_{\min}|$$

目标可以描述如下：

$$z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} \tag{8.22}$$

其中

$$c_{ij} = \alpha c^1_{ij} + \beta c^2_{ij}$$

图 8.3 给出了一个关于目标的解释。连接点($z^1_{\min}, z^2_{\max}$)和点($z^1_{\max}, z^2_{\min}$)的线将目标空间分为两部分：一部分包括正的理想解 Z^+，另一部分包括了负的理想解 Z^-。可行解空间 F 也相应地被分为两部分：$F^- = F \cap Z^-$ 和 $F^+ = F \cap Z^+$。容易证明 F^+ 中的解比 F^- 中的解具有较小的目标函数值(8.21)或(8.22)。

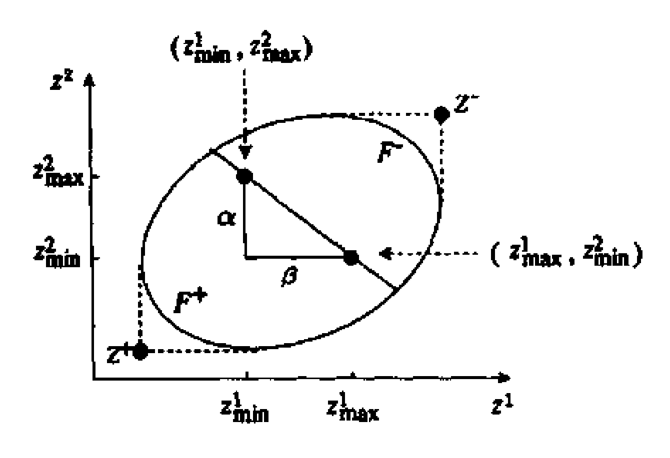


图 8.3 目标说明

对于每个染色体，适值由等式(8.21)决定。因而，在空间 F^+ 中的染色体更有可能进入下一代。在每一代中，修改集合 E 并更新两个特殊点。这意味着在进化过程中，连接点($z^1_{\min}, z^2_{\max}$)和点($z^1_{\max}, z^2_{\min}$)的线将沿着从负理想点向正理想点的方向逐渐地移动，即适值函数表明选择压力将迫使遗传搜索探索目标空间中的非劣点。

8.3.3 总体过程

Yang 和 Gen 方法的全部过程归纳如下：

Yang 和 Gen 求解 BLTP 的步骤

步骤 1：参数设置

设定种群大小 *pop_size*，变异率 p_m，交叉率 p_c 和最大代数 *max_gen*。令 $t \leftarrow \phi, E \leftarrow \varnothing$。

步骤 2：初始化

随机产生初始解矩阵 $X_p = (x^p_{ij})$，$p = 1, 2, \cdots,$ *pop_size*：

begin

for $p \leftarrow 1$ **to** *pop_size*

$\pi \leftarrow \{1, 2, \cdots, mn\}$;

repeat

从集合 π 中任选一个数值 k;

计算相应的行和列;

$i \leftarrow \lfloor (k-1)/m + 1 \rfloor$;

$j \leftarrow (k-1) \bmod m + 1$;

赋给 x_{ij}^{p} 可用量;

$x_{ij}^{p} \leftarrow \min\{a_i, b_j\}$;

修改数据;

$a_i \leftarrow a_i - x_{ij}^{p}$;

$b_j \leftarrow b_j - x_{ij}^{p}$;

$\pi \leftarrow \pi \backslash \{k\}$;

until π 为空

end

end

步骤 3：交叉

1）产生[0,1]之间的随机数 $r_p, p=1,2,\cdots, pop_size$。如果 $r_p < p_c$，则 X_p 选作交叉的双亲。

2）对于每对双亲 (X_p, X_q)，建立两个临时矩阵：

$$D = (d_{ij}), \quad R = (r_{ij})$$

其中

$$d_{ij} = \lfloor x_{ij}^{p} + x_{ij}^{q} \rfloor /2$$

$$r_{ij} = (x_{ij}^{p} + x_{ij}^{q}) \bmod 2$$

3）将矩阵 R 分成 $R^1 = (r_{ij}^1)$ 和 $R^2 = (r_{ij}^2)$ 两个矩阵，使它们满足：

$$R = R^1 + R^2$$

$$\sum_{j=1}^{n} r_{ij}^1 = \sum_{j=1}^{n} r_{ij}^2 = \frac{1}{2} \sum_{j=1}^{n} r_{ij}; \quad i = 1,2,\cdots,m$$

$$\sum_{i=1}^{m} r_{ij}^1 = \sum_{i=1}^{m} r_{ij}^2 = \frac{1}{2} \sum_{i=1}^{m} r_{ij}; \quad j = 1,2,\cdots,n$$

4）产生两个后代 X_p' 和 X_q'：

$$X_p' = D + R_1$$

$$X_q' = D + R_2$$

步骤 4：变异

1）产生[0,1]之间的随机数 $r_p, p = 1,2,\cdots,pop_size$。如果 $r_p < p_m$，则 X_p 选作变异双亲。

2）任选 p 行和 q 列，建立子矩阵 $Y = (y_{ij})_{p\times q}$。

3）子矩阵的物品可用量 a_i^y 和需求可用量 b_j^y 计算如下：

$$a_i^y = \sum_{j \in \{j_1, \cdots, j_q\}} y_{ij}; \quad i = i_1, i_2, \cdots, i_p$$

$$b_j^y = \sum_{i \in \{i_1, \cdots, i_p\}} y_{ij}; \quad j = j_1, j_2, \cdots, j_q$$

4）通过初始化过程对子矩阵重新分配物品，以满足所有 a_i^y 和 b_j^y 约束。令 Y' 代表新的子矩阵。

5）用子矩阵 Y' 中的新元素替换矩阵 X_p 中的适当元素，以产生后代 X_p'。

步骤 5：修改集合 E

1）双亲和后代 X_p 的每个染色体的双准则目标值计算如下：

$$z^p = (z^1, z^2)$$

其中

$$z^1 = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^1 x_{ij}$$

$$z^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^2 x_{ij}$$

2）通过向集合 E 中增加新的非劣点和从集合 E 中删除劣点 (dominated points)修改集合 E。

3）确定集合 E 中新的特殊点$(z_{\min}^1, z_{\max}^2)$和$(z_{\max}^1, z_{\min}^2)$：

$$z_{\min}^1 = \min \{z^1(\mathbf{x}) \mid \mathbf{x} \in E\}$$

$$z_{\max}^1 = \max \{z^1(\mathbf{x}) \mid \mathbf{x} \in E\}$$

$$z_{\min}^2 = \min \{z^2(\mathbf{x}) \mid \mathbf{x} \in E\}$$

$$z_{\max}^2 = \max \{z^2(\mathbf{x}) \mid \mathbf{x} \in E\}$$

步骤 6：评估

双亲和后代 X_p 每个染色体的适值计算如下：

$$eval(X_p) = \alpha z^1 + \beta z^2$$

其中

$$\alpha = |z_{\max}^2 - z_{\min}^2|$$

$$\beta = |z_{\max}^1 - z_{\min}^1|$$

步骤 7：选择

在双亲和后代中选择最好的 *pop_size* 个染色体以产生下一代。

步骤 8：终止检查

如果 $t = max_gen$，则停止；否则令 $t \leftarrow t+1$，转步骤 3。

8.3.4 数值实例

例 8.3 考虑 Aneja 和 Nair[8]给出的下述双准则线性运输问题：

$$\min \quad z^1 = x_{11} + 2x_{12} + 7x_{13} + 7x_{14} + x_{21} + 9x_{22} + 3x_{23} + 4x_{24} + 8x_{31} + 9x_{32} + 4x_{33} + 6x_{34}$$

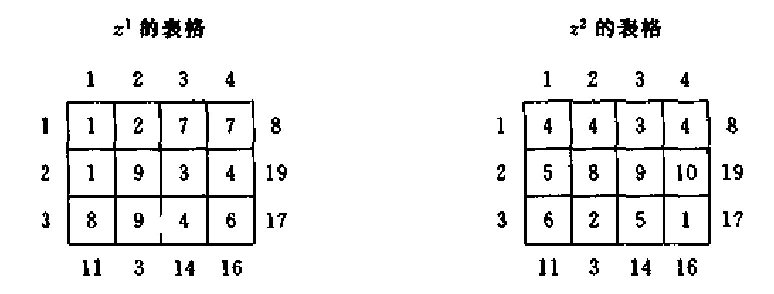
$$\min \quad z^2 = 4x_{11} + 4x_{12} + 3x_{13} + 4x_{14} + 5x_{21} + 8x_{22} + 9x_{23} + 10x_{24} + 6x_{31} + 2x_{32} + 5x_{33} + x_{34}$$

$$\text{s.t.} \quad \sum_{j=1}^{4} x_{1j} = 8, \sum_{j=1}^{4} x_{2j} = 19, \sum_{j=1}^{4} x_{3j} = 17$$

$$\sum_{i=1}^{3} x_{i1} = 11,\ \sum_{i=1}^{3} x_{i2} = 3,\ \sum_{i=1}^{3} x_{i3} = 14,\ \sum_{i=1}^{3} x_{i4} = 16$$

$$x_{ij} \geqslant 0;\quad \forall\, i,j$$

这一问题用表格描述如下：



z^1 的表格

	1	2	3	4	
1	1	2	7	7	8
2	1	9	3	4	19
3	8	9	4	6	17
	11	3	14	16	

z^2 的表格

	1	2	3	4	
1	4	4	3	4	8
2	5	8	9	10	19
3	6	2	5	1	17
	11	3	14	16	

种群大小定为 20，最大代数定为 1 000。根据 Michalewicz 的试验，交叉率小变异率大会获得更好的效果，确定变异率为 $p_m=0.2$，交叉率为 $p_c=0.4$。

我们运行五次，对于每次运行采用不同的随机数种子，每次运行的结果几乎是相同的，如图 8.4 所示。

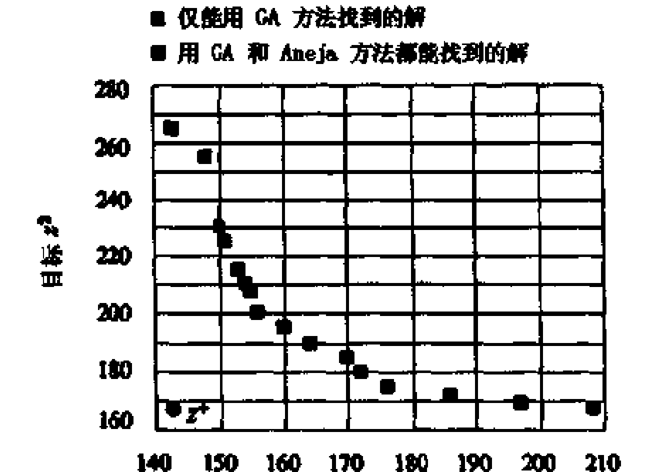


图 8.4　GA 获得的非劣解

从图 8.4 我们可以看出 GA 方法不仅可以找到文献[8,79]中给出的有效端点，而且还可以找到一些新的有效端点。

8.4　双准则三维运输问题

8.4.1　BSTP 的描述

三维运输问题 STP(Solid Transportation Problem)是基本运输问题的一般化。当存在多种运输工具运送物品时，就有必要考虑这类特殊的运输问题。三维运输问题经常用在公共分配系统并经常考虑一个以上的目标[36]。

假设存在 m 个起点(或源点)，n 个终点和 K 种运输工具。在每个起点，令 a_i 是要被运

送到 n 个终点的同种物品的数量，每个终点具有 b_j 个单位物品需求。令 e_k 是可用第 k 种运输工具运输的物品数量，对第 q 个目标函数的惩罚 $c_{ijk}^q, q=1,2$ 与用第 k 种运输工具从起点 i 到终点 j 运输一个单位的物品有关。惩罚可以代表运输费用、交货时间、物品交货质量、占用能力等等。变量 x_{ijk} 代表用第 k 种运输工具从起点 i 到终点 j 的运输量。双准则三维运输问题 BSTP 可描述如下：

$$\min \quad z^q = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{K} c_{ijk}^q x_{ijk}; \quad q=1,2 \tag{8.23}$$

$$\text{s.t.} \quad \sum_{j=1}^{n}\sum_{k=1}^{K} x_{ijk} = a_i; \quad i=1,2,\cdots,m \tag{8.24}$$

$$\sum_{i=1}^{m}\sum_{k=1}^{K} x_{ijk} = b_j; \quad j=1,2,\cdots,n \tag{8.25}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk} = e_k; \quad k=1,2,\cdots,K \tag{8.26}$$

$$x_{ijk} \geqslant 0; \quad \forall\, i,j,k \tag{8.27}$$

其中对于所有的 i 有 $a_i>0$，对于所有的 j 有 $b_j>0$，对于所有的 k 有 $e_k>0$，对于所有的 i，j，k，q 有 $c_{ijk}^q \geqslant 0$。上述公式假设总供应等于总需求，即

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = \sum_{k=1}^{K} e_k \quad \text{（平衡条件）} \tag{8.28}$$

平衡条件可作为问题存在可行解的充分必要条件。Gen 等进一步将他们的工作扩展到了 BSTP[155]。

8.4.2 初始化

对于三维运输问题，一个染色体可以用一个三维数组描述，也可以等价地用一个具有 nmK 个元素的矩阵描述：

$$[x_{111}x_{211}\cdots x_{m11},\cdots,x_{1n1}\ x_{2n1}\cdots x_{mn1},\cdots,x_{11K}x_{21K}\cdots x_{m1K},\cdots,x_{1nK}\ x_{2nK}\cdots x_{mnK}]$$

其中 x_{ijk} 是染色体中相应的决策变量。

我们可以很容易地将 LTP 的初始过程扩展到 STP 以适应三维情况。

初始化过程

begin

$\pi \leftarrow \{1, 2, \cdots, mnK\}$；

repeat

从集合 π 中任选一个数 l；

计算相应的下标；

$i \leftarrow (l-1) \bmod m + 1$；

$j \leftarrow \lfloor (l-1)/m \rfloor \bmod n + 1$；

$k \leftarrow \lfloor (l-1)/mn \rfloor + 1$；

赋给 x_{ijk} 可用量；

$x_{ijk} \leftarrow \min \{a_i, b_j, e_k\}$；

修改数据；

$a_i \leftarrow a_i - x_{ijk}$;

$b_j \leftarrow b_j - x_{ijk}$;

$e_k \leftarrow e_k - x_{ijk}$;

$\pi \leftarrow \pi \backslash \{l\}$;

until π 为空

end

8.4.3 遗传运算

LTP 的遗传因子可以类似地用于处理三维情况。

1. 交叉

假设两个双亲为 $X_1 = (x_{ijk}^1)$和 $X_2 = (x_{ijk}^2)$。交叉由下述三个步骤完成:

步骤 1:建立两个临时矩阵 $D = (d_{ijk})$和 $R = (r_{ijk})$如下:

$$d_{ijk} = \lfloor (x_{ijk}^1 + x_{ijk}^2) / 2 \rfloor \tag{8.29}$$

$$r_{ijk} = (x_{ijk}^1 + x_{ijk}^2) \bmod 2 \tag{8.30}$$

矩阵 D 存放两个双亲的取整均值,矩阵 R 记录是否需要取整。两个矩阵之间的关系由下述等式给出:

$$a_i - \sum_{j=1}^{n}\sum_{k=1}^{K} d_{ijk} = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{K} r_{ijk}; \quad i = 1,2,\cdots,m \tag{8.31}$$

$$b_j - \sum_{i=1}^{m}\sum_{k=1}^{K} d_{ijk} = \frac{1}{2}\sum_{i=1}^{m}\sum_{k=1}^{K} r_{ijk}; \quad j = 1,2,\cdots,n \tag{8.32}$$

$$e_k - \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ijk} = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ijk}; \quad k = 1,2,\cdots,K \tag{8.33}$$

步骤 2:将矩阵 R 分解成两个矩阵 $R^1 = (r_{ijk}^1)$和 $R^2 = (r_{ijk}^2)$使得

$$R = R^1 + R^2 \tag{8.34}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{K} r_{ijk}^1 = \sum_{j=1}^{n}\sum_{k=1}^{K} r_{ijk}^2 = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{K} r_{ijk}; \quad i = 1,2,\cdots,m \tag{8.35}$$

$$\sum_{i=1}^{m}\sum_{k=1}^{K} r_{ijk}^1 = \sum_{i=1}^{m}\sum_{k=1}^{K} r_{ijk}^2 = \frac{1}{2}\sum_{i=1}^{m}\sum_{k=1}^{K} r_{ijk}; \quad j = 1,2,\cdots,n \tag{8.36}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ijk}^1 = \sum_{i=1}^{m}\sum_{j=1}^{n} r_{ijk}^2 = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ijk}; \quad k = 1,2,\cdots,K \tag{8.37}$$

容易发现,存在许多在满足上述条件的情况下将 R 分解为 R^1 和 R^2 的方法。

步骤 3:产生两个后代 X_1' 和 X_2':

$$X_1' = D + R^1 \tag{8.38}$$

$$X_2' = D + R^2 \tag{8.39}$$

2. 变异

变异主要由以下三步完成:

步骤 1:由双亲建立一个子矩阵。任选$\{i_1,\cdots,i_p\}$,$\{j_1,\cdots,j_q\}$和$\{k_1,\cdots,k_s\}$建立一个

$(p*q*s)$的子矩阵 $Y=(y_{ijk})$，其中 $\{i_1,\cdots,i_p\}$ 是 $\{1,2,\cdots,m\}$ 的真子集，并且 $2\leqslant p\leqslant m$；$\{j_1,\cdots,j_q\}$ 是 $\{1,2,\cdots,n\}$ 的真子集，并且 $2\leqslant q\leqslant n$；$\{k_1,\cdots,k_s\}$ 是 $\{1,2,\cdots,K\}$ 的真子集，并且 $2\leqslant s\leqslant K$；y_{ijk} 从双亲矩阵中具有 ijk 下标的相应元素取值。

步骤 2：为子矩阵重新分配物品。子矩阵的物品可用量 a_i^y、需求可用量 b_j^y 和运输工具的能力 e_k^y 确定如下：

$$a_i^y=\sum_{j\in\{j_1,\cdots,j_q\}}\sum_{k\in\{k_1,\cdots,k_s\}}y_{ijk};\quad i=i_1,i_2,\cdots,i_p$$

$$b_j^y=\sum_{i\in\{i_1,\cdots,i_p\}}\sum_{k\in\{k_1,\cdots,k_s\}}y_{ijk};\quad j=j_1,j_2,\cdots,j_q$$

$$e_k^y=\sum_{i\in\{i_1,\cdots,i_p\}}\sum_{j\in\{j_1,\cdots,j_q\}}y_{ijk};\quad k=k_1,k_2,\cdots,k_s$$

我们可以用初始化过程为子矩阵赋新值使所有约束 a_i^y, b_j^y 和 e_k^y 得到满足。

步骤 3：用重新分配的子矩阵 Y 中的新元素替换双亲矩阵中的适当元素。

8.4.4 数值实例

例 8.4 考虑下述由 Bit，Biswal 和 Alam 给出的双准则三维运输问题[36]：

$$\begin{aligned}
\min\quad & z^q(X)=\sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{3}c_{ijk}^q x_{ijk};\quad q=1,2\\
\text{s.t.}\quad & \sum_{j=1}^{4}\sum_{k=1}^{3}x_{1jk}=24,\quad \sum_{j=1}^{4}\sum_{k=1}^{3}x_{2jk}=8\\
& \sum_{j=1}^{4}\sum_{k=1}^{3}x_{3jk}=18,\quad \sum_{j=1}^{4}\sum_{k=1}^{3}x_{4jk}=10\\
& \sum_{i=1}^{4}\sum_{k=1}^{3}x_{i1k}=11,\quad \sum_{i=1}^{4}\sum_{k=1}^{3}x_{i2k}=19\\
& \sum_{i=1}^{4}\sum_{k=1}^{3}x_{i3k}=21,\quad \sum_{i=1}^{4}\sum_{k=1}^{3}x_{i4k}=9\\
& \sum_{i=1}^{4}\sum_{j=1}^{4}x_{ij1}=17,\quad \sum_{i=1}^{4}\sum_{j=1}^{4}x_{ij2}=31\\
& \sum_{i=1}^{4}\sum_{j=1}^{4}x_{ij3}=12\\
& x_{ijk}\geqslant 0,\quad \forall\, i,j,k
\end{aligned}$$

$$C^1=\begin{array}{c}\\1\\2\\3\\4\end{array}\begin{array}{c}\begin{array}{cccccccccccc}c_{i11}&c_{i12}&c_{i13}&c_{i21}&c_{i22}&c_{i23}&c_{i31}&c_{i32}&c_{i33}&c_{i41}&c_{i42}&c_{i43}\end{array}\\ \left[\begin{array}{cccccccccccc}15&18&17&12&22&13&10&4&12&8&11&13\\17&20&19&21&21&22&21&19&18&30&10&23\\14&11&12&25&34&33&20&16&15&21&23&22\\22&18&13&24&35&32&18&21&14&13&23&20\end{array}\right]\end{array}$$

$$C^2 = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \begin{bmatrix} 6 & 7 & 8 & 10 & 6 & 5 & 11 & 3 & 7 & 10 & 9 & 6 \\ 13 & 8 & 11 & 12 & 2 & 9 & 20 & 15 & 13 & 17 & 15 & 13 \\ 5 & 6 & 7 & 11 & 9 & 7 & 10 & 5 & 2 & 15 & 14 & 18 \\ 13 & 6 & 6 & 17 & 11 & 18 & 12 & 16 & 12 & 18 & 14 & 7 \end{bmatrix}$$

$pop_size=20, p_m=0.2, p_c=0.4, max_gen=1\ 000$。1 000代后，我们得到下述 29 个非劣解：

(703, 537) (710, 418) (715, 394) (733, 376)
(752, 360) (778, 355) (784, 349) (793, 343)
(796, 330) (800, 320) (805, 316) (810, 314)
(811, 313) (814, 312) (815, 310) (816, 309)
(817, 308) (819, 307) (820, 306) (823, 305)
(824, 304) (825, 303) (826, 302) (836, 300)
(837, 299) (838, 298) (846, 296) (852, 295)
(866, 293)

这些解用图 8.5 描述。

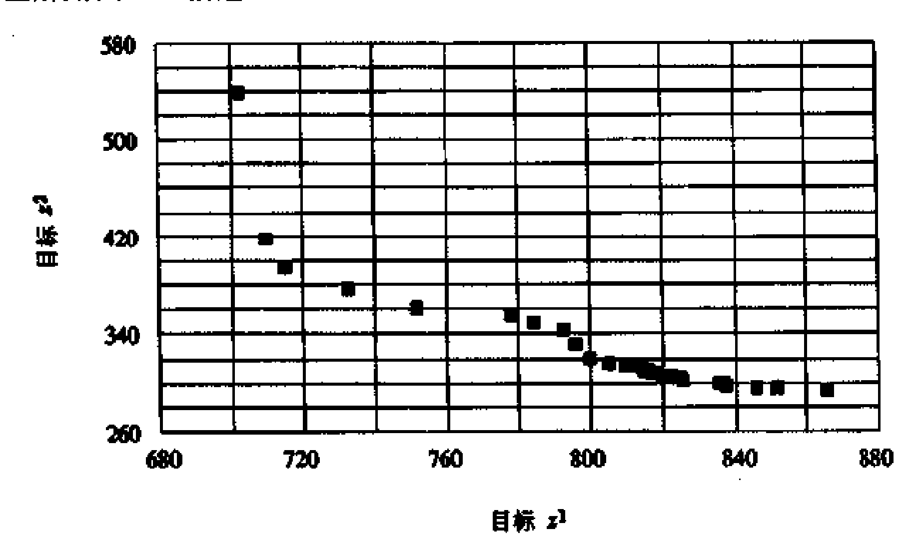


图 8.5 用 GA 获得的非劣解

8.5 模糊多准则三维运输问题

8.5.1 问题描述

采用多目标方法，当制定运输计划时，除了运输费用，还考虑许多其它有影响的因素，如交货时间、交货产品质量、运输占用能力可靠性、用户可访问性和产品损失等。在现实情况下，由于社会和经济环境的复杂性以及一些不可预测的因素(如天气)，共同的问题是难以确定适当的模型参数。处理这类决策中不确定性的一种方法就是模糊规划[58,466]。Kaufmann 和 Gupta 最先研究了模糊运输问题[244]。最近，Bit，Biswal 和 Alam 作了一些相关研究[35,36]，他们用模糊规划技术处理了多准则运输问题，而不是模糊运输问题。Gen，Ida 和 Li 研究了下述模糊多准则三维运输问题[157]：

$$\min \quad \tilde{z}_q = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{K} \tilde{c}_{ijk}^{q} x_{ijk}; \quad q = 1,2,\cdots,Q \tag{8.40}$$

$$\text{s.t.} \quad \sum_{j=1}^{n}\sum_{k=1}^{K} x_{ijk} = a_i; \quad i = 1,2,\cdots,m \tag{8.41}$$

$$\sum_{i=1}^{m}\sum_{k=1}^{K} x_{ijk} = b_j; \quad j = 1,2,\cdots,n \tag{8.42}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk} = e_k; \quad k = 1,2,\cdots,K \tag{8.43}$$

$$x_{ijk} \geqslant 0; \quad \forall\ i,j,k$$

其中对于所有 i 有 $a_i \geqslant 0$，对于所有 j 有 $b_j \geqslant 0$，对于所有 k 有 $e_k \geqslant 0$。约束包括前一节讨论的平衡条件。唯一的不同是目标函数的系数是用模糊数 $\tilde{c}_{ijk}^{q}, q=1,2,\cdots,Q$ 表示的。

8.5.2 遗传算法

Gen，Ida 和 Li 提出了一种解决模糊多目标三维运输问题的遗传算法[157,451]。基本实现与前一节给出的方法相同，重点是处理模糊性。在多目标优化中，我们主要是寻找 Pareto 解。当目标函数的系数是用模糊数表示时，目标值也成为了模糊数。由于模糊数代表许多可能的实数，对比各个解确定 Pareto 解是不容易的。模糊排序技术可帮助我们对比模糊数[58]。在 Gen 的方法中，Pareto 解用模糊目标的顺序值来确定，而用遗传算法搜索 Pareto 解。

1. 用积分排序模糊数

Liou 和 Wang 提出了一种用积分值为模糊数排序的方法[276]。一般地，隶属函数可分为左右两部分[114,432]。对于极小化问题，左部分反函数的积分值反映决策者的乐观观点，右部分反函数的积分值反映决策者的悲观观点。用右部分和左部分积分值的乐观指标的凸组合（称作全局积分值）可对模糊数排序。

令 $\tilde{A}$ 是三角模糊数，由 (a_1,a_2,a_3) 表示。它的隶属函数 $\mu_{\tilde{A}}$ 由下式给出：

$$\mu_{\tilde{A}}(x) = \begin{cases} (x - a_1)/(a_2 - a_1), & a_1 \leqslant x \leqslant a_2 \\ (x - a_3)/(a_2 - a_3), & a_2 \leqslant x \leqslant a_3 \\ 0, & \text{其他} \end{cases} \tag{8.44}$$

其中 a_1，a_2 和 a_3 是实数，见图 8.6。左右隶属函数分别用 $\mu_{\tilde{A}}(x)^L=(x-a_1)/(a_2-a_1)$ 和 $\mu_{\tilde{A}}(x)^R=(x-a_3)/(a_2-a_3)$ 表示。

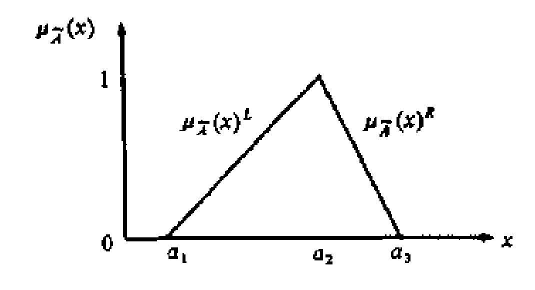


图 8.6 隶属函数 $\mu_{\tilde{A}}$

$\mu_{\tilde{A}}(x)^L$ 和 $\mu_{\tilde{A}}(x)^R$ 相应的反函数可表述如下：

$$g_{\tilde{A}}(y)^L = a_1 + (a_2 - a_1)y \tag{8.45}$$

$$g_{\tilde{A}}(y)^R = a_3 + (a_2 - a_3)y \tag{8.46}$$

因而左右积分值为

$$I(\tilde{A})^L = \int_0^1 g_{\tilde{A}}(y)^L dy = \frac{1}{2}(a_1 + a_2) \tag{8.47}$$

$$I(\tilde{A})^R = \int_0^1 g_{\tilde{A}}(y)^R dy = \frac{1}{2}(a_2 + a_3) \tag{8.48}$$

当参数 $\alpha \in [0,1]$用来调整决策者的乐观程度时，三角模糊数(TFN)$\tilde{A}$ 的全局积分值为

$$I^{\alpha}(\tilde{A}) = \alpha\, I(\tilde{A})^R + (1-\alpha)I(\tilde{A})^L = \frac{1}{2}[\alpha\, a_3 + a_2 + (1-\alpha)a_1] \tag{8.49}$$

当决策乐观程度 α 为 0.5 时，上述积分值与通常的描述相同[244]。

例如，两个三角模糊数 $\tilde{A}_1 = (3,4,7)$和 $\tilde{A}_2 = \left(4,5,\frac{51}{8}\right)$的全局积分值是

$$I^{\alpha}(\tilde{A}_1) = 4 + 2\,\alpha \text{ 和 } I^{\alpha}(\tilde{A}_2) = 4.5 + 1.2\,\alpha$$

在最小化问题中，对于 $\alpha=1$ 的悲观决策者，排出的顺序为 $\tilde{A}_1 > \tilde{A}_2$；对于 $\alpha=0$ 的乐观决策者，排出的顺序为 $\tilde{A}_1 < \tilde{A}_2$；对于 $\alpha=0.5$ 的中等的决策者排出的顺序为 $\tilde{A}_1 > \tilde{A}_2$。

采用这一排序方法，模糊目标值可转换为积分值，Pareto 解可根据这些积分值来确定。

2. 评估

在多目标情况下如何评估每个染色体的适值是遗传算法评估阶段相对困难的任务。在 Gen，Ida 和 Li 的方法中，用加权和的方法计算染色体的适值。其过程如下：

评估过程

步骤 1：对每个染色体 X_s，$s=1,2,\cdots,pop_size$，计算模糊目标值 $\tilde{z}_q(X_s)$，$q=1,2,\cdots,Q$；

步骤 2：用排序方法将模糊数 $\tilde{z}_q(X_s)$转换为积分值 $I_q^{\alpha}(\tilde{z}_q(X_s))$；

步骤 3：每个目标的最大和最小积分值确定如下：

$$I_q^{\min} = \min_s\{I_q^{\alpha}(\tilde{z}_q(X_s));\ s=1,2,\cdots,pop_size\};\quad q=1,2,\cdots,Q$$

$$I_q^{\max} = \max_s\{I_q^{\alpha}(\tilde{z}_q(X_s));\ s=1,2,\cdots,pop_size\};\quad q=1,2,\cdots,Q$$

步骤 4：加权系数计算如下：

$$\delta_q = I_q^{\max} - I_q^{\min};\quad q=1,2,\cdots,Q$$

$$\beta_q = \frac{\delta_q}{\sum_{q=1}^{Q}\delta_q};\quad q=1,2,\cdots,Q$$

步骤 5：每个染色体的适值计算如下：

$$eval(X_s) = \sum_{q=1}^{Q}\beta_q I_q^{\alpha}(\tilde{z}_q(X_s));\quad s=1,2,\cdots,pop_size$$

3. 最好折衷解

将 Yoon 和 Hwang 给出的 TOPSIS 方法嵌入 GA 方法，可用来确定在 Pareto 解中的最好折衷(compromise)解。TOPSIS 代表类似理想解的优先级(order preference)技术，这项技术的基本思想是应该选择与正理想解距离最短、与负理想解距离最长的解[255]。令 $X_k, k=1,2,\cdots,m$ 是遗传算法最后一代的 Pareto 解，则 TOPSIS 方法如下：

TOPSIS 方法步骤

步骤 1：确定近似正、负理想解

$$I_q^+ = \min\{I_q^a(\tilde{z}_q(X_s)); s = 1,2,\cdots,m\}; \quad q = 1,2,\cdots,Q$$

$$I_q^- = \max\{I_q^a(\tilde{z}_q(X_s)); s = 1,2,\cdots,m\}; \quad q = 1,2,\cdots,Q$$

步骤 2：建立标准值

$$r_q^k = \frac{I_q^a(\tilde{z}_q(X_k))}{\sqrt{\sum_{s=1}^{m}(I_q^a(\tilde{z}_q(X_s)))^2 + (I_q^+)^2 + (I_q^-)^2}}$$

步骤 3：计算分离指标

$$s_k^+ = \sqrt{\sum_{q=1}^{Q} w_q^2(r_q^+ - r_q^k)^2}; \quad k = 1,2,\cdots,m$$

$$s_k^- = \sqrt{\sum_{q=1}^{Q} w_q^2(r_q^k - r_q^-)^2}; \quad k = 1,2,\cdots,m$$

其中 r_q^+ 和 r_q^- 是标准化的 I_q^+ 和 I_q^-，w_q 是满足下述条件的目标相对权重：

$$\sum_{q=1}^{Q} w_q = 1,, w_q \in [0,1]$$

步骤 4：计算理想解的相对靠近度

$$d_k = \frac{s_k^-}{s_k^+ + s_k^-}; \quad k = 1,2,\cdots, m$$

步骤 5：优先级排序

Pareto 解可按 d_k 的递减顺序来排序。

模糊多准则三维运输问题步骤

begin

$t \leftarrow 0$;

初始化 $P(t)$;

评估 $P(t)$;

建立 Pareto 解 $E(t)$;

while 非终止条件 **do**

重组 $P(t)$;

评估 $P(t)$;

选择 $P(t+1)$;

修改 Pareto 解 $E(t)$;

$t \leftarrow t+1$;

end

确定最好折衷解

end

8.5.3 数值实例

考虑下面实例：

$$\min \quad \tilde{z}_q = \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} \tilde{c}^q_{ijk} x_{ijk}; \quad q = 1,2,3$$

$$\text{s.t.} \quad \sum_{j=1}^{3}\sum_{k=1}^{3} x_{1jk} = 8, \quad \sum_{j=1}^{3}\sum_{k=1}^{3} x_{2jk} = 9, \quad \sum_{j=1}^{3}\sum_{k=1}^{3} x_{3jk} = 5$$

$$\sum_{i=1}^{3}\sum_{k=1}^{3} x_{i1k} = 7, \quad \sum_{i=1}^{3}\sum_{k=1}^{3} x_{i2k} = 6, \quad \sum_{i=1}^{3}\sum_{k=1}^{3} x_{i2k} = 5$$

$$\sum_{i=1}^{3}\sum_{j=1}^{3} x_{ij1} = 10, \quad \sum_{i=1}^{3}\sum_{j=1}^{3} x_{ij1} = 5, \quad \sum_{i=1}^{3}\sum_{j=1}^{3} x_{ij1} = 6$$

$$x_{ijk} \geqslant 0; \quad \forall i,j,k$$

这是根据 Bit，Biswal 和 Alam 的例子改编的[36]。表 8.1 是模糊系数 $\tilde{c}^q_{ijk}$。

表 8.1 目标函数的模糊系数

q	1			2		
k	1	2	3	1	2	3
$\tilde{c}^q_{11k}$	(8,9,10)	(10,12,14)	(7,9,11)	(3,6,9)	(8,9,10)	(5,7,9)
$\tilde{c}^q_{12k}$	(4,5,6)	(5,6,7)	(3,5,7)	(7,9,11)	(8,11,14)	(1,3,5)
$\tilde{c}^q_{13k}$	(1,2,3)	(1,2,3)	(1,1,1)	(1,2,3)	(6,7,8)	(6,7,8)
$\tilde{c}^q_{21k}$	(1,2,3)	(8,9,10)	(6,8,10)	(1,1,1)	(2,4,6)	(1,1,1)
$\tilde{c}^q_{22k}$	(1,2,3)	(7, 8,9)	(1,1,1)	(3,4,5)	(3,5,7)	(1,2,3)
$\tilde{c}^q_{23k}$	(4, 5, 6)	(1,2,3)	(5,7,9)	(6,8,10)	(8,9,10)	(6,7,8)
$\tilde{c}^q_{31k}$	(1,2,3)	(2,4,6)	(5,6,7)	(2,3,4)	(4,6,8)	(3,4,5)
$\tilde{c}^q_{32k}$	(1,2,3)	(3, 5,7)	(1,3,5)	(3,5,7)	(4,6,8)	(4,6,8)
$\tilde{c}^q_{33k}$	(1,1,1)	(8,9,10)	(1,1,1)	(7,8,9)	(2,3,4)	(7,9,11)

q	3		
k	1	2	3
$\tilde{c}^q_{11k}$	(2,3,4)	(6,7,8)	(5,7,9)
$\tilde{c}^q_{12k}$	(5,6,7)	(6,8,10)	(5,6,7)
$\tilde{c}^q_{13k}$	(1,1,1)	(8,9,10)	(1,3,5)
$\tilde{c}^q_{21k}$	(7,9,11)	(7,9,11)	(4,5,6)
$\tilde{c}^q_{22k}$	(6,8,10)	(5,6,7)	(8,9,10)
$\tilde{c}^q_{23k}$	(3,5,7)	(1,2,3)	(3,5,7)
$\tilde{c}^q_{31k}$	(6;8,10)	(2,4,6)	(7,9,11)
$\tilde{c}^q_{32k}$	(7,9,11)	(4,6,8)	(1,3,5)
$\tilde{c}^q_{33k}$	(3,5,7)	(5,7,9)	(10,11,12)

试验中参数确定如下：$max_gen=1\ 000$，$pop_size=30$，$p_m=0.2$，$p_c=0.4$，$w_1=0.5$，$w_2=0.3$，$w_3=0.2$。通过10次运行，对于乐观和适中情况下的Pareto解在表8.2中给出。

表8.2 获得Pareto最优解

乐观情况($\alpha=0$)				
序号	$\tilde{z}_1$	$\tilde{z}_2$	$\tilde{z}_3$	$I=(I_1,I_2,I_3)$
1	(67,100,133)	(97,123,149)	(82,120,158)	(83.5,110,101)
2	(75,98,121)	(87,112,137)	(70,107,139)	(86.5,99.5,88.5)
3	(71,103,135)	(78,106,134)	(70,107,144)	(87,92,91)
4*	(75,114,153)	(50,65,80)	(48,86,124)	(94.5,57.5,67)
5	(89,129,169)	(95,113,131)	(52,78,104)	(109,104,65)
6	(89,130,171)	(48,63,78)	(90,126,162)	(109.5,55.5,108)
7	(103,141,179)	(43,59,75)	(83,118,153)	(122, 51, 100.5)
8	(109,139,169)	(115,143,171)	(47,78,109)	(124,129, 62.5)
9	(106,143,180)	(42,56,70)	(61,94,127)	(124.5,49,77.5)
10	(117,152,187)	(38,55,72)	(76,110,144)	(134.5,46.5,93)
11	(134,158,182)	(116,139,162)	(40,71,102)	(146,127.5,55.5)
12	(136,159,182)	(114,138,162)	(47,78,109)	(147.5,126,62.5)
$I^+=(83.5, 46.5, 55.5)$			$I^-=(147, 129, 108)$	

适中情况($\alpha=0.5$)				
序号	$\tilde{z}_1$	$\tilde{z}_2$	$\tilde{z}_3$	$I=(I_1,I_2,I_3)$
1	(73,78,123)	(132,162,192)	(105,147,189)	(98,162,147)
2	(74,103,132)	(100,130,160)	(86,125,164)	(103,130,125)
3	(71,104,137)	(94,121,148)	(80,118,156)	(104,121,118)
4*	(73,109,145)	(53,71,89)	(48,87,126)	(109,71,87)
5	(75,114,153)	(50,65,80)	(48,86,124)	(114,65,86)
6	(81,122,163)	(46,59,72)	(60,98,136)	(122,59,98)
7	(128,153,178)	(125,149,173)	(41,74,107)	(153,149,74)
8	(130,156,182)	(100,125,150)	(36,69,102)	(156,125,69)
$I^+=(98, 59, 69)$			$I^-=(156,162,147)$	

每种情况最好的折衷解用*标识。乐观情况($\alpha=0$)下的解是

$$x_{121}=6,\quad x_{331}=4,\quad x_{132}=2$$

$$x_{232}=2,\quad x_{332}=1,\quad x_{213}=7$$

适中情况($\alpha=0.5$)下的解是

$$x_{121}=5,\quad x_{331}=5,\quad x_{122}=1$$

$$x_{132}=2,\quad x_{232}=2,\quad x_{213}=7$$

第九章　设备布局设计问题

9.1 引　　言

二十多年来，设备布局设计一直是跨学科的研究课题。设备布局的发展过程中既包含艺术因素又包含科学的因素。最近这一课题得到运筹学和管理科学工作者的极大注意。它经常被看作是优化问题，最优设备布局在满足一些约束条件下可通过优化某些性能指标得到[138]。

迄今为止，制造技术取得了重大进展，在世界范围内实现了大量的柔性制造系统(FMS)，制造系统物理布局优化设计是在系统设计初期必须解决的重要问题之一。布局决策的好处可以在系统实现和系统运行过程中体现出来。关于这些问题的好的解决方案为有效地利用系统提供了必要的基础。但是获得这样的解决方案是一项复杂的任务。布局设计者面临着的艰巨任务是以合理价格开发一个系统，使它能处理具有需求变化的各种产品。

在制造环境中，布局设计是指在确定的区域内最适宜地安排物理设备，如车间或机器。通常设计准则是最小化物料储运费用。由于设备布局问题的组合特性，启发式技术是解决实际规模布局问题的最有效方法。最近，用遗传算法解决设备布局设计的趋势迅速加强。Tate 和 Smith 将遗传算法用于有形状约束的不等区域设备布局问题[399]。Cohoon 等提出解决车间布局设计问题的分布式的遗传算法[81]。Tam 介绍了将遗传算法用于设备布局问题的经验[395]。在这一章中，我们将介绍如何能成功地将遗传算法应用于不同的机器布局和设备布局问题。

9.2　机器布局问题

柔性机器系统是实现计算机辅助制造的一种形式。柔性机器系统的设计包括机器和工作站的布局。Kusiak 和 Heragu 发表了研究机器布局问题的文章[259]。在柔性机器系统中机器的布局通常是由所采用的物料储运设备类型决定的。经常采用的物料储运设备包括：

(1) 物料储运机器人；

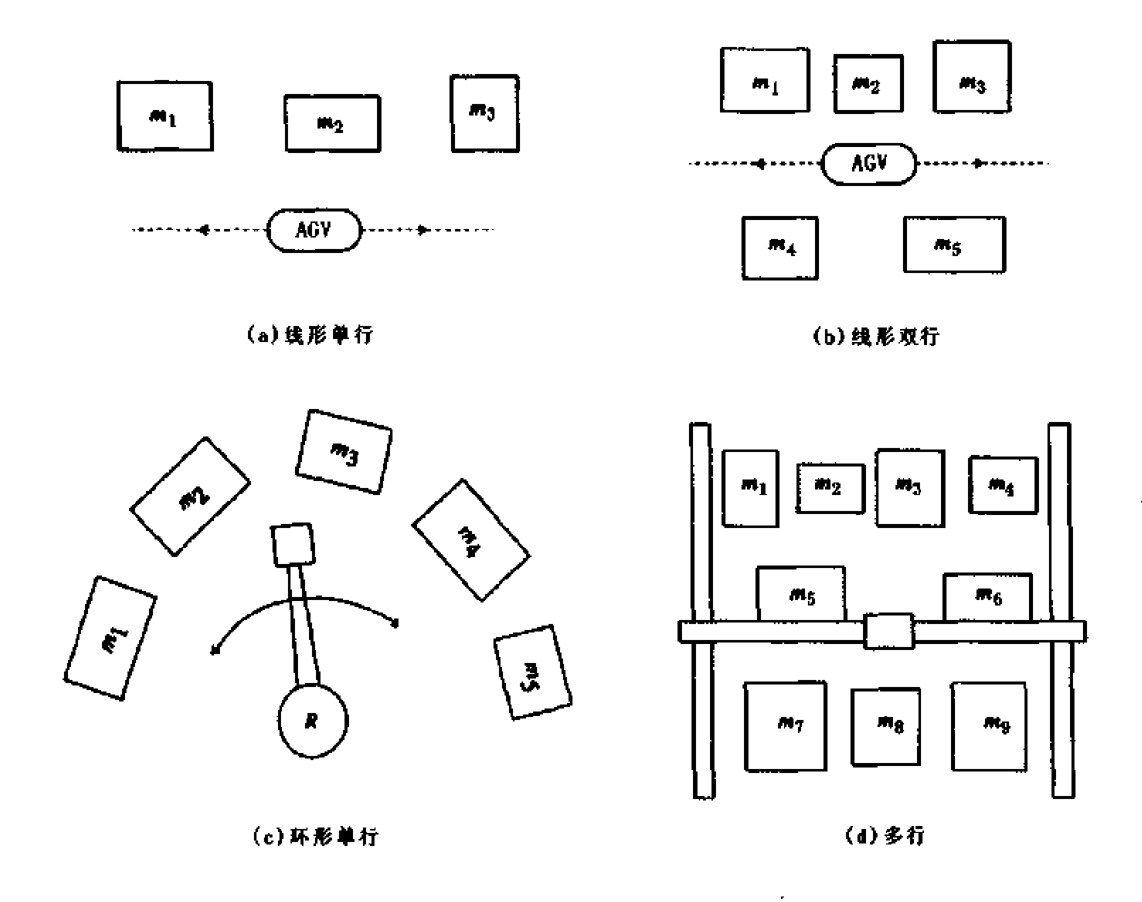
(2) 自动引导车(AGV)；

(3) 门式机器人。

图 9.1 给出了实际中经常遇到的机器布局的四种基本类型。由一个 AGV 服务的机器趋于沿着直线安排[图 9.1 (a)和(b)]，而机器人的可达到范围约束使机器排成环形(circle)或簇状，见图 9.1 (c)和(d)。

机器布局的四种基本类型命名如下：

(1) 线形单行；



(a) 线形单行　(b) 线形双行

(c) 环形单行　(d) 多行

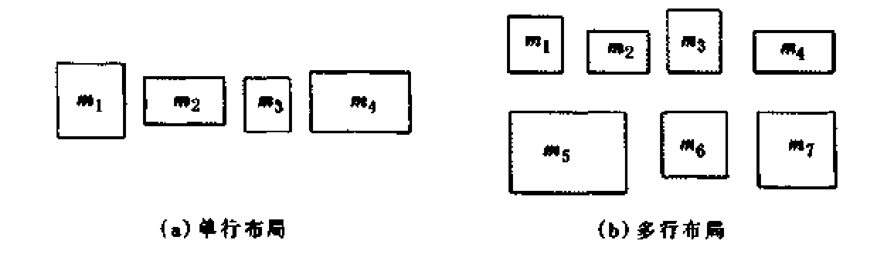
图 9.1　机器布局的基本类型

(2) 线形双行；

(3) 环形单行；

(4) 多行。

为了对布局问题建模，只需考虑单行模式和多行模式。因为在图 9.1 的四种布局类型中，环形单行和线形单行可以看作单行布局模式；线形双行布局是多行布局的一个特例。在单行模式中，机器安排在一行内；而在多行模式中，机器成线形地安排在两行或多行内。图 9.2 中是一些简单的实例。



(a) 单行布局　(b) 多行布局

图 9.2　机器布局模式

描述布局设计问题的大多数常用方法是将其作为二次分配问题，这种方法需预先说明每个点的位置，然后为每个点指派设备以使全部的物料储运费用最小。Heragu 和 Kusi-

ak 指出柔性制造系统中的机器布局与传统的设备布局问题在一个重要的方面不同[211]，即机器的大小通常不同。如果两台机器间的距离由各自的尺寸和它们之间需要的间距决定，则位置之间的距离是不等的并且是不可预先确定的。为此，他们提出了在目标函数和约束中含有绝对值的新的机器布局问题模型。

9.3 单行机器布局问题

9.3.1 数学模型

为了对单行机器布局问题建模，作如下假设[258]：

(1) 机器是矩形的；

(2) 机器方位已知；例如，所有的机器都是纵向放置的。

单行机器布局问题可以描述如下：

$$\min \quad \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij} f_{ij} |x_i - x_j| \tag{9.1}$$

$$\text{s.t.} \quad |x_i - x_j| \geqslant \frac{1}{2}(l_i + l_j) + d_{ij}; \quad i = 1,\cdots,n-1; j = i+1,\cdots,n \tag{9.2}$$

$$x_i \geqslant 0; \quad i = 1,\cdots,n \tag{9.3}$$

其中：

n——机器数；

f_{ij}——机器 i 和机器 j 之间的访问频率；

c_{ij}——机器 i 和机器 j 之间访问时每单位距离的储运费用；

l_i——机器 i 的长度；

d_{ij}——机器 i 和机器 j 之间的最小间距；

x_i——机器 i 的中心线相对垂直参考线 l_V 的距离。

参数 l_i 和 d_{ij}、决策变量 x_i 和垂直参考线 l_V 如图 9.3 所示。约束(9.2)确保任何两台机器都不会出现布局重叠。约束(9.3)确保非负性。目标是最小化机器之间作必要次数访问时的总费用。

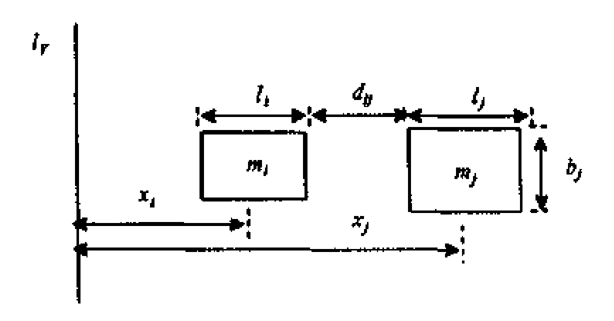


图 9.3 参数和决策变量示意图

9.3.2 单行机器布局问题的遗传算法

单行机器布局是一种非常特殊的情况，这一问题可看作是机器的排序问题，因而它可以由两步完成：

(1) 机器排序；

(2) 产生实际布局。

第一步产生机器排序，第二步根据机器的排序和几何需求建立实际布局。正如我们所知，遗传算法对于处理这样的排列问题是非常有效的。本节中给出的算法的基本思想是采用遗传算法寻找一个较好的机器排序，然后，根据机器的排序和几何需求建立实际布局。

1. 表达方式

对于单行情况，机器的排序换位是对机器布局编码以获得染色体的一种直接方法。例如，机器按下列顺序安排：

$$[m_2 \quad m_1 \quad m_4 \quad m_6 \quad m_3 \quad m_5]$$

其中 m_i 代表第 i 个机器，则染色体 $\mathbf{v}_k$ 表示为

$$\mathbf{v}_k = [2\ 1\ 4\ 6\ 3\ 5]$$

对于这样的编码，存在许多可用的遗传算子，我们采用部分映射交叉法(PMX)和反转变异法进行交叉和变异。细节见第三章。

2. 评估

通常对于 n 台机器问题，染色体 $\mathbf{v}_k$ 给定如下：

$$\mathbf{v}_k = [m_1^k,\ m_2^k,\ \cdots,\ m_n^k]$$

其中 m_i^k 代表在第 k 个染色体第 i 个位置的机器(或机器符号)。根据机器的排序和几何尺寸需求我们可以计算所有机器的 x 轴坐标(机器的位置或机器的实际布局)：

$$[x_1^k,\ x_2^k,\ \cdots,\ x_n^k]$$

然后我们可以计算第 k 个染色体的总费用：

$$f_k = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij} f_{ij} |x_i^k - x_j^k| \tag{9.4}$$

由于布局设计问题是最小化问题，我们必须将每个染色体的目标函数值转换成适值，以使适宜的染色体具有较大的适值。转换由下述评估函数完成：

$$eval(\mathbf{v}_k) = \frac{1}{f_k} \tag{9.5}$$

其中，$eval(\mathbf{v}_k)$是第 k 个染色体的适值函数。

用转轮法作产生下一代的基本选择机制。在这个阶段采用了精选策略，以保护下一代最好染色体，并克服取样的随机错误。

3. 实例

试验问题是由 Kusiak 给出的一个六台机器的布局问题[258]。机器大小、频率矩阵、费用矩阵和机器间的间距如下：

机器大小

机器 i	尺寸($l_i \times b_i$)
1	5.0×3.0
2	2.0×2.0
3	2.5×2.0
4	6.0×3.5
5	3.0×1.5
6	4.0×4.0

$$[f_{ij}]=\begin{bmatrix} 0 & 40 & 80 & 21 & 62 & 90 \\ 40 & 0 & 72 & 12 & 24 & 28 \\ 80 & 72 & 0 & 14 & 41 & 9 \\ 21 & 12 & 14 & 0 & 21 & 12 \\ 62 & 24 & 41 & 21 & 0 & 31 \\ 90 & 28 & 9 & 12 & 31 & 0 \end{bmatrix}$$

$$[c_{ij}]=\begin{bmatrix} 0 & 4 & 4 & 6 & 4 & 5 \\ 4 & 0 & 2 & 5 & 2 & 3 \\ 4 & 2 & 0 & 5 & 3 & 3 \\ 6 & 5 & 5 & 0 & 5 & 8 \\ 4 & 2 & 3 & 5 & 0 & 4 \\ 5 & 3 & 3 & 8 & 4 & 0 \end{bmatrix}$$

$$[d_{ij}]=\begin{bmatrix} 0 & 1 & 1 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 3 & 1 \\ 2 & 1 & 1 & 3 & 0 & 2 \\ 1 & 1 & 1 & 1 & 2 & 0 \end{bmatrix}$$

（各矩阵的行、列编号均为 1～6。）

进化环境是：$pop_size=20$，$max_gen=50$，$p_c=0.4$，$p_m=0.2$。最好染色体如下：

最好解产生的代数：17

费用：19 531.00

机器排序：[6 1 3 5 2 4]

基于排序的布局(见图 9.4 所示)，进化过程如图 9.5 所示。

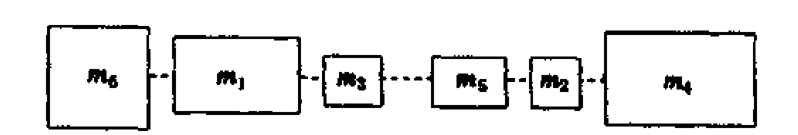


图 9.4 实例问题的单行布局

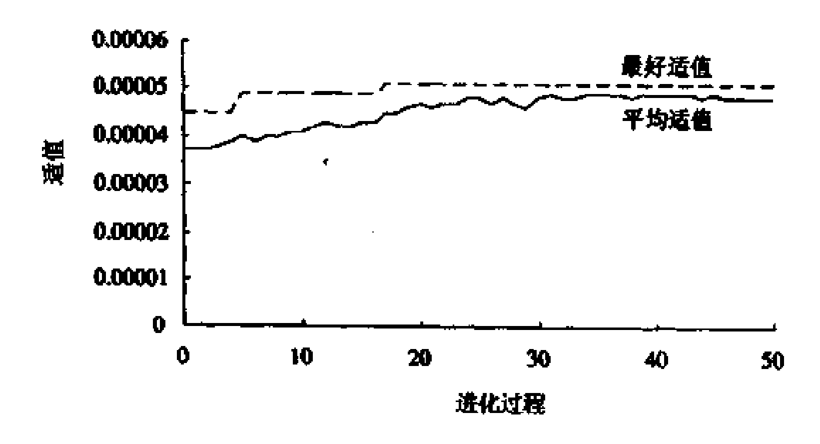


图 9.5 实例问题的进化过程

9.4 多行机器布局问题

9.4.1 数学模型

解决设备布局问题的许多传统方法是将设备分配给事先确定的位置，这是离散优化方法。但在机器布局问题中，位置不能事先确定。Heragu 和 Kusiak 将多行机器布局问题描述为属于连续优化方法的二维连续空间分配问题[211]。然而，在许多实际问题中，机器安排在事先定义好的行内，这是因为许多情况下，行的分隔可根据物料储运系统的特点事先确定，即这一问题在一维上看作是离散的，在另一维上看作是连续的。令 x_i 和 y_i 分别代表从机器 i 的中心线到垂直和水平的参考线的距离，令决策变量 z_{ik} 为

$$z_{ik} = \begin{cases} 1, & \text{如果机器 } i \text{ 分配给行 } k \\ 0, & \text{其他} \end{cases} \tag{9.6}$$

具有不等区域的多行机器布局问题可以描述为混和整数规划问题：

$$\min \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij}(|x_i - x_j| + |y_i - y_j|) \tag{9.7}$$

$$\text{s.t.} \quad |x_i - x_j| z_{ik} z_{jk} \geqslant \frac{1}{2}(l_i + l_j) + d_{ij}; \quad i,j = 1, \cdots, n \tag{9.8}$$

$$y_i = \sum_{k=1}^{m} l_0(k-1) z_{ik}; \quad i = 1, \cdots, n \tag{9.9}$$

$$\sum_{k=1}^{m} z_{ik} = 1; \quad i = 1, \cdots, n \tag{9.10}$$

$$\sum_{i=1}^{n} z_{ik} \leqslant n; \quad k = 1, \cdots, m \tag{9.11}$$

$$x_i, y_i \geqslant 0; \quad i = 1, \cdots, n \tag{9.12}$$

$$z_{ik} = 0, 1; \quad i = 1, \cdots, n, k = 1, \cdots, m \tag{9.13}$$

其中：

n——机器的数量；

m——行数；

f_{ij}——机器 i 和机器 j 之间的访问频率；

c_{ij}——机器 i 和机器 j 之间访问时每单位距离的储运费用；

l_i——机器 i 的长度；

l_0——两个相邻行的中心距；

d_{ij}——机器 i 和机器 j 之间的最小间距；

x_i——机器 i 中心与垂直参考线 l_V 之间的距离；

y_i——机器 i 中心与水平参考线 l_H 之间的距离。

目标是极小化机器间作必要次数访问时的总费用，约束(9.8)确保任何两台机器均不重叠，约束(9.10)～(9.11)确保一台机器只放在一行上。由约束(9.9)可知模型中的变量 y_i 是不必要的，因为它可以由 z_{ik}确定，但这一约束有助于理解模型。参数、决策变量和参考线如图 9.6 所示。

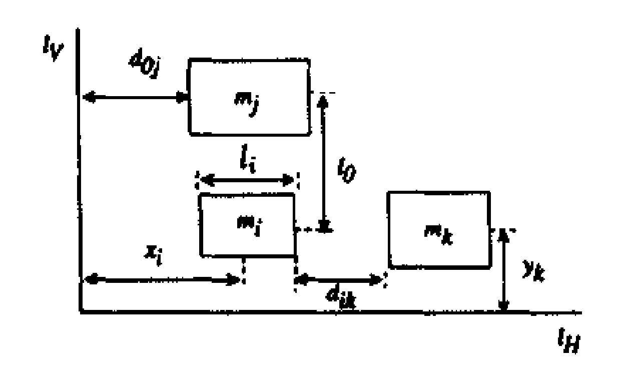


图 9.6 参数、决策变量和参考线示意图

从该模型我们可以看出这一问题主要包括两项任务：

(1) 分配机器到行(确定 y 坐标)；

(2) 在每行内为机器寻找最佳位置(确定 x 坐标)。

第一项任务是组合优化问题，尽管第二项任务是单行布局问题，但由于存在行间访问费用，每行最优解不是问题的全局最优解。因而，我们不能简单地将这一问题当作多个单行布局问题来处理。

9.4.2 表达方式

对于多行机器布局问题，表达方法在文献[59,67]中给出，这一描述机制可以看作是包括分隔符、机器符号、净间距三个列表的扩展换位表达方式。机器符号序列代表机器所有可能的换位(或机器排序)，分隔符代表机器在行上的分配，净间距序列记录所有机器的 x 轴位置。对于 n 台机器和两行的情况，表达方式为

$$[s, \{m_{i_1}, m_{i_2}, \cdots, m_{i_n}\}, \{\Delta_{i_1}, \Delta_{i_2}, \cdots, \Delta_{i_n}\}]$$

其中 m_{i_j} 代表在第 j 个位置的机器，Δ_{i_j} 代表机器 $m_{i_{j-1}}$ 和机器 m_{i_j} 间的净间距。分隔符 s 通常满足 $1<s<n$，代表序列被分成两部分的分隔位置。第一部分中的机器分配在第一行，其他机器分配在第二行。

这一描述机制属于直接编码方法，我们很自然会想到一个问题，为什么不直接对 x 轴和 y 轴的值进行染色体编码？因为这一特殊方法有许多优势。

(1) 首先，这一编码技术可以显著减少进化过程中不合理后代的产生。这一表达方式中，x 轴坐标由净间距表示，而不是它的绝对值。如果我们在编码表达方式中采用 x 轴坐标，有可能产生大量在水平方向上机器重叠的不合理后代。采用净间距编码，机器重叠将不会发生。同样，为避免纵向机器重叠，我们用分隔符序列和机器排列序列替代对 y 轴值编码的表达机制，以表达行间的机器分配。

(2) 其次，我们知道，实质上多行机器布局问题的关键是离散和连续优化问题的组合，即

1) 沿 x 轴的连续优化问题；

2) 沿 y 轴的组合优化问题。

当进行布局时，行间机器分配和每行内的机器换位是主要的因素，对由分配和排序决定的机器安排的细微调整也要受到机器的位置影响。采用这一编码机制，我们可以分三部

分处理这一棘手问题：

(1) 分配机器到行；

(2) 在行内对机器排序；

(3) 细微调整每台机器的位置(x 和 y 坐标)；

(4) 最后，因为我们采用序列表达机器排列，所以可以采用一些用于表达排序的传统的遗传算子来处理这一问题。

可见，所提出的上述表达方法与问题的本质相吻合，并使遗传算法非常有效。

1. 计算 x 轴坐标

x 轴坐标可根据净间距序列和机器排列序列唯一确定。假设机器 m_k 和机器 m_i 排在同一行内，如图 9.7 所示。

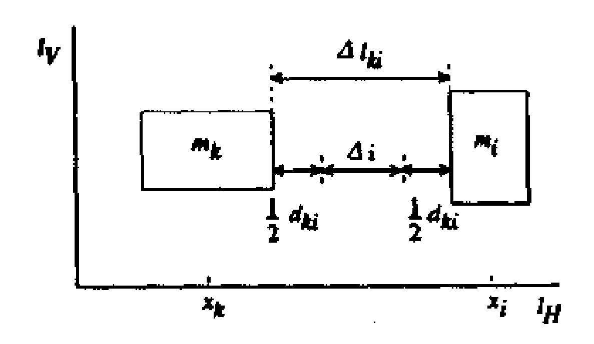


图 9.7 净间距和决策变量示意图

净间距和决策变量计算如下：

$$\Delta_i = \Delta l_{ki} - d_{ki}$$

$$x_i = x_k + d_{ki} + \Delta_i + \frac{1}{2}(l_i + l_k)$$

$$x_k = d_{k0} + \Delta_k + \frac{1}{2}l_k$$

其中：

Δ_i——机器 m_i 的净间距；

Δl_{ki}——两台机器间的间距；

d_{ki}——机器 k 和机器 i 之间的必须间距。

从上图我们可以看出两个相邻机器间的间距由两部分组成：必须间距和净间距。由于净间距将会带来机器间访问总费用的增加，因此人们常认为，所有机器间净间距为零是最好的。对于单行情况，这一结论是成立的，但是，对于多行情况，由于存在行间访问费用，这一结论是不成立的。

2. 计算 y 轴坐标

y 轴坐标可基于行间距确定，而行间距可根据物料储运系统特点事先确定。考虑两行情况，令 l 代表两行间的距离，如图 9.8 所示。

假设第一行的位置为 0，y 轴坐标计算如下：

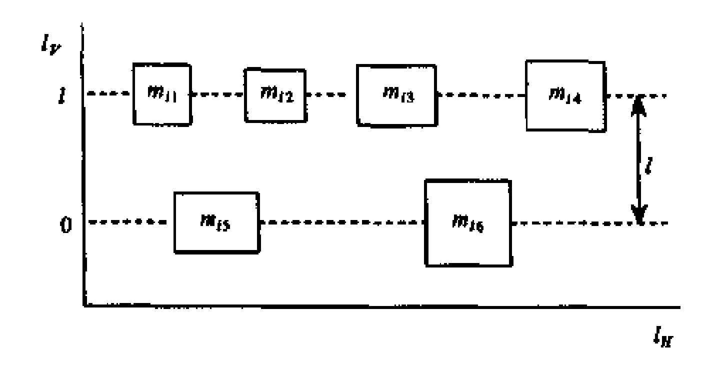


图 9.8　y 坐标示意图

$$y_i = \begin{cases} 0, & \text{如果 } m_i \text{ 在第一行} \\ l, & \text{如果 } m_i \text{ 在第二行} \end{cases}$$

现在我们用一个实例介绍如何用这一直接编码机制建立实际的布局。假设我们有如下的染色体：

[4，{3，6，1，5，4，2 }，{1.2，0.2，0.4，0.3，2.2，4.6 }]

根据这一编码，实际布局如图 9.9 所示。

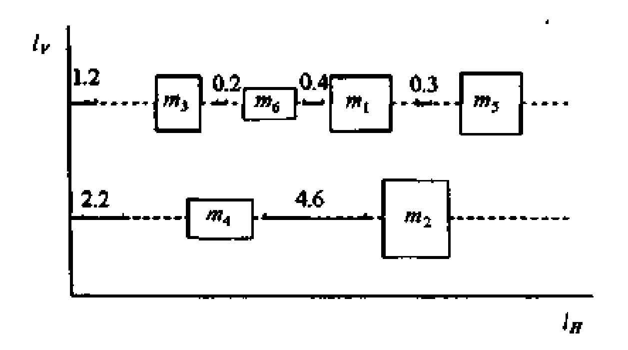


图 9.9　实际机器布局

9.4.3　初始化

采用随机方法产生初始种群：分隔符、机器排列序列、净间距序列。全部过程如下：

初始化过程

bigin

　$i \leftarrow 0$;

　while $i \leqslant pop_size$ **do**

　　生成分隔符；

　　生成机器排序序列；

　　生成净间距序列；

　　$i \leftarrow i+1$;

　end

end

1. 分隔符

对于 n 台机器、两行的情况，分隔符只是开区间 $(1, n)$ 上的任意整数。

2. 机器换位

令 $\sum_0$ 是可用机器的集合，P 是机器换位序列，则机器顺序由下述过程随机产生：

机器换位过程

begin

  $i \leftarrow 0$;

  $\sum_0 \leftarrow \{1, 2, \cdots, n\}$;

  $P \leftarrow \phi$;

  while $i \leqslant n$ **do**

    从 $\sum_0$ 中任取机器 m'；

    $P \leftarrow P \cup m'$;

    $\sum_0 \leftarrow \sum_0 / m'$;

    $i \leftarrow i+1$;

  end

  返回排列序列 P；

end

3. 净间距

产生净间距序列的过程实质上是一个随机方法。由于存在允许空间约束，净间距在允许区域内产生。令 L' 是可用空间，L 是允许工作区域的上限宽度，假设机器序列如下：

$$[\, m_1 \quad m_2 \quad \cdots \quad m_n \,]$$

则初始可用空间计算如下：

$$L' = 2L - \left(\sum_{i=1}^{n} l_i + \sum_{i=1}^{n-1} d_{i,i+1} + d_{10} + d_{n0} \right) \tag{9.14}$$

令 Δ 表示净间距序列，整个过程如下：

净间距过程

begin

  $i \leftarrow 0$;

  $\Delta \leftarrow \phi$;

  计算初始可用空间 L'；

  while $i \leqslant n$ **do**

    在 $[0, L']$ 上任取净间距 Δ_i；

    $\Delta \leftarrow \Delta \cup \Delta_i$;

    $L' \leftarrow L' - \Delta_i$;

    $i \leftarrow i+2$;

```
    end
    返回净间距序列 Δ;
  end
```

9.4.4 交叉

所提出的交叉的基本元素包括三个部分：随机方法确定分隔符；PMX 方法处理机器排列序列；算术交叉处理净间距序列。交叉过程描述如下：

```
交叉过程
begin
  i←0;
  while i ≤ pop_size × p_c do
    任选两个个体;
    产生一个新的分隔符;
    用 PMX 方法产生新的机器排列;
    采用算术交叉方法产生新的净间距序列;
    i←i+1;
  end
end
```

例如，有两个双亲：

$$p_1 = [\, s^1,\ \{ m_1^1,\ m_2^1,\ \cdots,\ m_n^1\},\ \{\Delta_1^1,\ \Delta_2^1,\ \cdots,\ \Delta_n^1\} \,]$$

$$p_2 = [\, s^2,\ \{ m_1^2,\ m_2^2,\ \cdots,\ m_n^2\},\ \{\Delta_1^2,\ \Delta_2^2,\ \cdots,\ \Delta_n^2\} \,]$$

现在我们介绍如何产生一个后代。

1. 新的分隔符

产生新的分隔符的过程很简单，主要由两步组成：

(1) 确定分隔符的上下限；

(2) 从上下限构成的区间内任选一个整数。

上下限可直接进行计算：

$$s_U = \max\ \{s^1,\ s^2\}$$

$$s_L = \min\ \{s^1,\ s^2\}$$

然后用 s_U 和 s_L 构造闭区间$[s_L,\ s_U]$，新的分隔符是这一区间内的任意整数。

2. 新的净间距序列

我们定义一种新的算术交叉来处理净间距序列。假设存在如下两个净间距序列：

$$\{\Delta_1^1,\ \Delta_2^1,\ \cdots,\ \Delta_n^1\}$$

$$\{\Delta_1^2,\ \Delta_2^2,\ \cdots,\ \Delta_n^2\}$$

每个机器新的净间距可计算如下：

$$\Delta_i' = a_1 \Delta_i^1 + a_2 \Delta_i^2; \quad i = 1,\ 2,\ \cdots,\ n \tag{9.15}$$

$$\alpha_1, \alpha_2 \in (0, 1) \tag{9.16}$$

所提出的交叉与常用算术交叉的主要不同在于常用交叉需满足：

$$\alpha_1 + \alpha_2 = 1 \tag{9.17}$$

而所提出的交叉需满足：

$$\alpha_1 + \alpha_2 \leqslant 2 \tag{9.18}$$

其原因在于如果我们采用通常方法，则产生的净间距将随着进化过程逐渐减小。这种情况下搜索空间在很大程度上依赖于初始解空间。如果我们使参数满足不等式(9.18)，则搜索空间可以大大增大，并且独立于初始搜索空间。

9.4.5 变异

变异采用邻域搜索技术对机器位置进行细微调整。

1. 产生邻域

假设给定染色体的净间距序列是

$$\{\Delta_1, \Delta_2, \cdots, \Delta_i, \cdots, \Delta_n\}$$

选非零基因，如第 i 个基因 Δ_i 进行变异，令 r 是一个给定的整数，则我们可以得到 $2r$ 个净间距：

$$\Delta_i^1 = \frac{\Delta_i}{r} \tag{9.19}$$

$$\Delta_i^j = \Delta_i^{j-1} + \frac{\Delta_i}{r}; \quad j = 2, 3, \cdots, 2r \tag{9.20}$$

这些净间距从 Δ_i/r 到 $2\Delta_i$ 变化。净间距序列集合如下：

$$\{\Delta_1, \Delta_2, \cdots, \Delta_i^1, \cdots, \Delta_n\}$$
$$\{\Delta_1, \Delta_2, \cdots, \Delta_i^2, \cdots, \Delta_n\}$$
$$\cdots$$
$$\{\Delta_1, \Delta_2, \cdots, \Delta_i^j, \cdots, \Delta_n\}$$
$$\cdots$$
$$\{\Delta_1, \Delta_2, \cdots, \Delta_i^{2r}, \cdots, \Delta_n\}$$

由上述净间距序列集合、给定染色体的分隔符及机器换位序列组成的染色体集合可看作为给定染色体的邻域。如果一个染色体比邻域中的其他任何染色体都好，则被称作 $2r$ 优化。

2. 变异过程

变异过程由三步完成，首先选择变异染色体，并从选定染色体中任选一个净间距；其次，生成选定染色体的 $2r$ 邻域；最后，评估所有邻域，并将最好邻域作为后代。这里提出的变异机制描述如下：

变异过程

begin

　给定一个整数 r；

```
    i←0;
    while i ≤ pop_size × p_m do
        任选一个染色体 v';
        从 v' 的净间距序列中任选一个基因 Δ_j;
        产生 v' 的 2r 邻域;
        评估所有邻域;
        选择最好邻域作为后代;
        i← i+1;
    end
end
```

9.4.6 评估函数

评估函数有两项:总费用和不合理惩罚。总费用用目标函数计算,令 v_k 是扩大种群中的第 k 个染色体:

$$[\ s^k,\ \{\ m_1^k,\ m_2^k,\ \cdots,\ m_n^k\},\ \{\Delta_1^k,\ \Delta_2^k,\ \cdots,\ \Delta_n^k\}\ ]$$

其中,m_j^k 是第 j 个位置上的机器;Δ_j^k 代表机器 m_{j-1}和机器 m_j 间的净间距;分隔符 s_k 通常是 $1<s_k<n$,代表根据两行需求将序列分成两部分的分隔位置。根据上述给出的过程,我们可以计算所有机器的 x 轴和 y 轴的坐标(机器位置或实际机器布局):

$$\{(x_1^k,\ y_1^k),\ (x_2^k,\ y_2^k),\ \cdots,\ (x_n^k, y_n^k)\}$$

最后我们得到第 k 个染色体的总费用如下:

$$f_k = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij} f_{ij}(|x_i^k - x_j^k| + |\ y_i^k - y_j^k|) \tag{9.21}$$

通常,机器布局问题中会产生两类不合理解:

(1) 机器重叠;

(2) 超出工作区域。

由于 x 轴坐标描述为净间距,在这一编码机制中,重叠不合理解不会发生。超出工作区域按以下方法确定:对于给定染色体 v_k,令 L_k^1 和 L_k^2 分别是安排在第一行和第二行的机器需要的必要工作区域,令 $L_k^u=\max\ \{L_k^1,\ L_k^2\}$,则惩罚系数 λ_k 计算如下:

$$\lambda_k = \begin{cases} 0, & 若\ L_u^k - L \leqslant 0 \\ L_k^u - L, & 其他 \end{cases} \tag{9.22}$$

评估函数可表示为

$$eval(v_k) = \frac{1}{f_k + \lambda_k P}; \qquad k = 1,2,\ \cdots,\ pop_size \tag{9.23}$$

其中,P 是正的大数惩罚值;f_k 是总费用。评估完每个染色体后,我们以转轮法作为选择机制在当前代的基础上繁殖下一代。

9.4.7 实例

试验问题包括六台机器,频率矩阵和费用矩阵与 9.3.2 节中单行情况下的相同,机器间的间距为

$$
\begin{array}{c} \\ [d_{ij}] = \end{array}
\begin{array}{c} d_{10} \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad d_{60} \\
\begin{bmatrix} 2 & 0 & 1 & 1 & 1 & 2 & 1 & 2 \\ 2 & 1 & 0 & 1 & 1 & 1 & 1 & 2 \\ 2 & 1 & 1 & 0 & 1 & 1 & 1 & 2 \\ 2 & 1 & 1 & 1 & 0 & 3 & 1 & 2 \\ 2 & 2 & 1 & 1 & 3 & 0 & 2 & 2 \\ 2 & 1 & 1 & 1 & 1 & 2 & 0 & 2 \end{bmatrix} \end{array} \tag{9.24}
$$

两行间的中心距为 8，工作区域的宽度约束是 22。进化环境如下：$pop_size=20$，$max_gen=500$，$p_c=0.4$，$p_m=0.4$，$r=10$。最好染色体如下：

最好解产生的代数： 48
适值： 0.5217
第一行的机器： 2 3 1 6
第二行的机器： 4 5
第一行机器的位置： 4.49，7.74，12.49，17.99
第二行机器的位置： 5.00，12.5

基于排序的布局如图 9.10 所示，进化过程如图 9.11 所示。

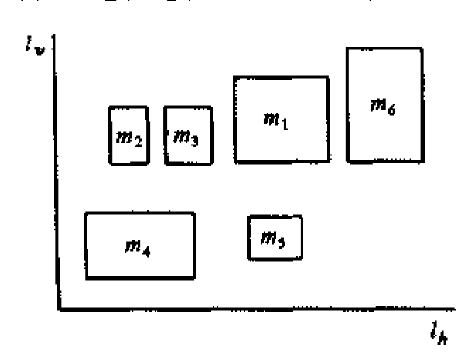


图 9.10 实例问题的多行布局

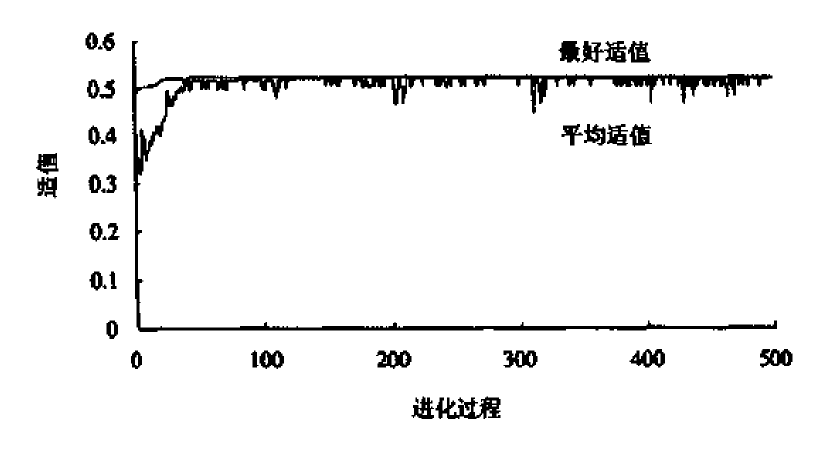


图 9.11 实例问题的进化过程

9.5 模糊设备布局问题

在设备布局问题的许多传统描述中假设设备之间的流量是已知的固定数量，并且设备布局问题在计划展望期上看作是静态问题[259]。然而，这一确定假设不符合现代制造系统。由于经营、发展、需求波动以及生产混和的动态特性，设备之间的流量随着阶段的不同而变化。如果一些这样的改变是可计划的，则是可以事先知道的，因而可以用 Rosenblatt 的动态布局模型来处理这一变化[357]。遗憾的是，产品混和、机器故障、周期性的波动和需求实际上是不确定的。在这种情况下，设计者希望得到的是满意布局，而不是最优布局。

对这一不确定性建模已提出两种方法。一种是柔性或概率方法[380]，另一种是鲁棒性方法[252，358]。如果采用柔性方法，我们必须确定物流的概率分布；如果采用鲁棒性方法，我们必须提供一些需求情况及每种情况的最优解。正如我们所知，物理布局最优设计必须在制造系统设计的早期解决，并且，在设计阶段，生产混和很不确定。给出准确的概率分布或给出一些准确的需求情况与设备布局优化设计本身一样困难。在这种情况下，设备之间物流估计很大程度上依赖于管理者或设计者的主观判断和（或）偏好，这实质上是不准确的。

Cheng 和 Gen 提出了处理这种不确定性的一种方法——模糊方法[70]。模糊集理论是对产生于心理现象的不确定性和不准确性进行建模的有效方法。当可用信息不准确并且有关不确定性不能忽略时，它处理不确定性和不准确性的能力是关键的[336]。主观或偏好信息通常（不总是）可描述为凸模糊数[325]，因而，我们可以用模糊数学处理这类不确定性和不准确性。模糊方法的实际价值在于不必强迫管理者或设计者给出准确的生产混和或概率分布。

9.5.1 设备布局问题

设备布局问题描述如下[420]：有一个 m 台设备的集合，表示为 $\{M_i\}$，$i=1, 2 \cdots, m$。每台设备是矩形的，并且由三元组 (A_i, l_i, u_i) 描述，其中 A_i 是设备面积，l_i 和 u_i 分别是外观比率上下限。外观比率定义为设备高 h_i 对设备宽 w_i 的比率。这一描述可导出下述关系：

$$w_i h_i = A_i;\quad i = 1, 2, \cdots, m \tag{9.25}$$

$$l_i \leqslant \frac{h_i}{w_i} \leqslant u_i;\quad i = 1, 2 \cdots, m \tag{9.26}$$

外观比率也可定义为它的倒数，图 9.12 给出了设备的高、宽和外观比率的关系。

给定 m 台设备的设备布局在有界矩形 R 上进行，R 被水平和垂直线段分割成 m 个不重叠的矩形区域，用 $\{r_i\}$，$i=1, 2, \cdots, m$ 表示。具有 x_i 宽和 y_i 高的每个区域 r_i 必须足够大，以装下设备 M_i。

9.5.2 模糊混和

通常，设备间物流不确定可描述为被称作模糊混和（fuzzy interflow）的凸模糊数，这

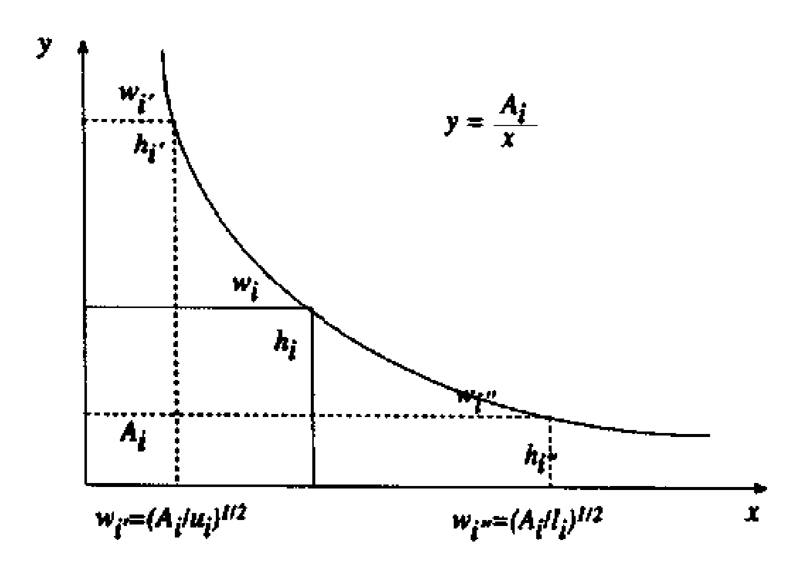


图 9.12　设备的高、宽和外观比率的关系

里采用梯形模糊数(TrFN)描述模糊混和[243]。一个梯形模糊数 TrFN 可以由四元组(a,b,c,d)定义，如图 9.13 所示。

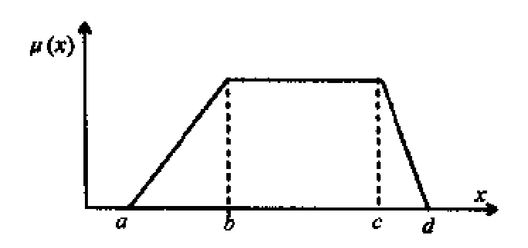


图 9.13　梯形模糊数

由于一个模糊数代表了具有不同隶属度的许多可能实数值，因而难以比较最终顺序以决定哪一个选择更好。人们已提出许多比较模糊数的模糊排序方法[56]。Lee 和 Li 的方法建议用基于模糊事件概率指标的均值和标准偏差为模糊数排序[267]。当模糊数 $\tilde{M}$ 是梯形模糊数时，具有均匀密度的均值计算如下：

$$m(\tilde{M}) = \frac{-a^2b^2 + c^2 + d^2 - ab + cd}{3(-a-b+c+d)} \tag{9.27}$$

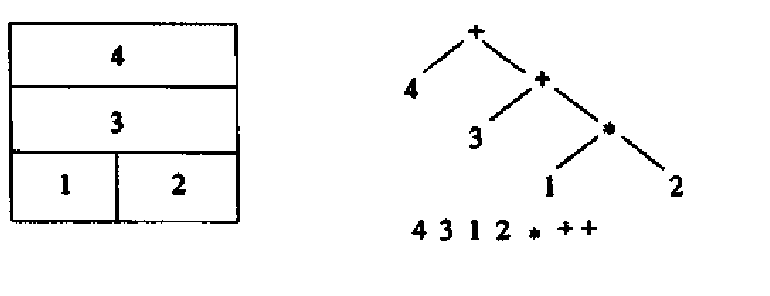
标准差定义如下：

$$\sigma(\tilde{M}) = \left[\frac{1}{b-a}\left(\frac{b^4}{4} - \frac{ab^4}{3} + \frac{a^4}{12}\right) + \frac{1}{3}(c^3 - b^3) + \frac{1}{d-c}\left(\frac{d^4}{12} - \frac{c^3d}{3} + \frac{c^4}{4}\right)\right] \Big/ \left[\frac{1}{2}(-a-b+c+d)\right] - m(\tilde{M})^2 \tag{9.28}$$

9.5.3　表达方式

设备布局可通过递归地分隔矩形 R 所构成的分割结构来表达。令水平切割和垂直切割分别由算子＋和 * 表示，分隔结构包括 m 个给定的设备(称作操作数)，它可以用字母集 $\sum = \{1, 2, \cdots, m, *, +\}$ 的分割树或波兰表达法(Polish expressions)表达。在分割树上，算子是内部节点，操作数是叶子。图 9.14(a)是分割结构的一个例子，它的分割

树和波兰表达法如图 9.14(b)所示。



(a) 设备布局实例　　(b) 分割树和波兰表示

图 9.14　分割结构及其表示

在遗传算法中用波兰表达法作为染色体的编码机制，一个染色体必须有 m 个不同的操作数和 $(m-1)$ 个算子，其中 m 是设备的个数。

9.5.4　初始化

建议采用有一定导向性的随机方法对遗传系统进行初始化。从左向右每次从集合 $\sum$ 中选出一个元素构造染色体，直到染色体构成为止。因为操作数和算子的随机排列将产生不合法染色体，所以，每次随机选择，都要检查它的合法性。

首先，我们定义染色体前缀的概念，对于大小为 $(2m-1)$ 的给定染色体，前缀是包括染色体的前 i 个元素的部分表述（不是合法的波兰描述），它具有与所在染色体中相同的顺序，其中 $i \leqslant 2m-1$。图 9.15 是一个实例。

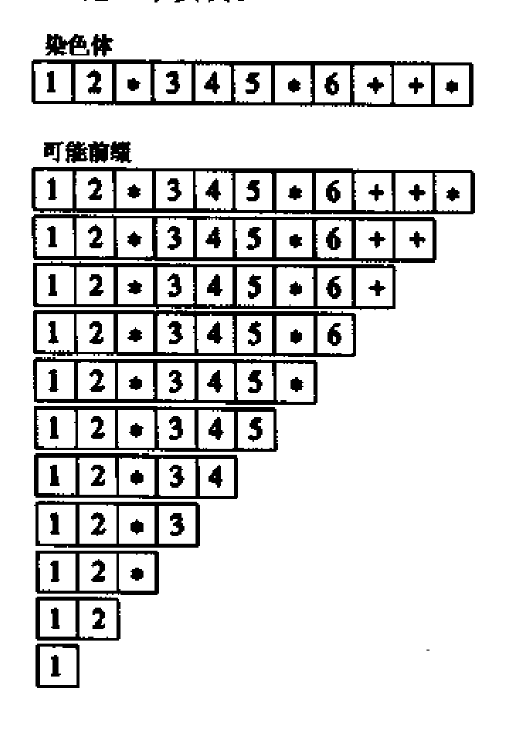


图 9.15　波兰前缀

令 Ψ 表示给定染色体所有可能前缀的集合，对于前缀 $\psi \in \Psi$，令 $N(\psi)$ 表示前缀 ψ 的操作数的总数，$M(\psi)$ 表示前缀 ψ 的算子的总数。染色体（或波兰表达）的合法性条件由下述

命题和推论给出。

命题 9.1 对于给定的含有 m 个操作数和 $(m-1)$ 个算子的染色体，如果等式

$$N(\psi) \geqslant M(\psi) + 1, \qquad \forall \psi \in \Psi \tag{9.29}$$

成立，则染色体是一个合法的波兰表达。

容易证明图 9.15 给出的染色体是合法的。

推论 9.1 对于给定前缀 ψ，如果不符合公式(9.29)的条件，则不可能从前缀 ψ 通过自左向右的生成过程建立合法的波兰表达。

例如前缀(1 2 * 3 4)满足公式(9.29)，因此，我们可以通过一个自左向右的生成过程由它建立合法的波兰表达：(1 2 * 3 4 + +)；前缀(1 2 * + +)不符合等式(9.29)，因而，我们不能从它通过自左向右的生成过程建立合法的波兰表达。

初始过程从集合 $\sum = \{1, 2, \cdots, m, *, +\}$ 中自左向右每次任选一个元素构造染色体，直到完整的染色体形成。每次选择都进行合法检查。假设我们有一个前缀(或部分产生的染色体)ψ，根据推论 9.1，如果 $N(\psi) = M(\psi) + 1$，要选择的下一元素必须是操作数；否则，要选择的下一元素可以是操作数，也可以是算子。整个过程如下：

有导向性的随机初始化过程

begin
  $\sum_1 \leftarrow \{*, +\}$;
  $i \leftarrow 1$;
  while $i \leqslant pop_size$ **do**
    $\sum_0 \leftarrow \{1, 2, \cdots, m\}$;
    $\psi_i \leftarrow \phi$;
    从 $\sum_0$ 中任选一元素 σ_1;
    $\sum_0 \leftarrow \sum_0 / \{\sigma_1\}$;
    $\psi_i \leftarrow \psi_i \cup \{\sigma_1\}$;
    $j \leftarrow 1$;
    while $j \leqslant 2m-1$ **do**
      if $N(\psi_i) = M(\psi_i) + 1$ **then**
        从 $\sum_0$ 中选 σ_j;
      else
        从 $\sum_0 \cup \sum_1$ 中选 σ_j;
      end
      $\sum_0 \leftarrow \sum_0 / \{\sigma_j\}$;
      $\psi_i \leftarrow \psi_i \cup \{\sigma_j\}$;
      $j \leftarrow j+1$;
    end
    $i \leftarrow i+1$;
  end
end

9.5.5 交叉

这里采用的交叉算子是由 Cohoon 等提出的[81]。交叉需两步进行：

(1) 从一个双亲获得操作数；

(2) 从其他双亲获得算子。

假设我们有两个双亲 p_1 和 p_2，交叉算子从双亲 p_1 得到操作数，放在后代 o 相应的位置；然后从双亲 p_2 获得算子，通过从左向右的搜索，完成后代 o。交叉这样进行可保证产生合法后代。图 9.16 说明了交叉操作，图 9.17 给出了双亲和后代的布局。

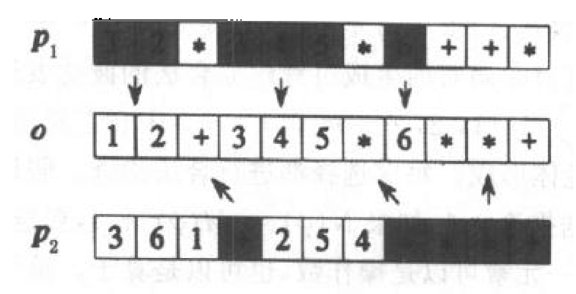


图 9.16 交叉算子

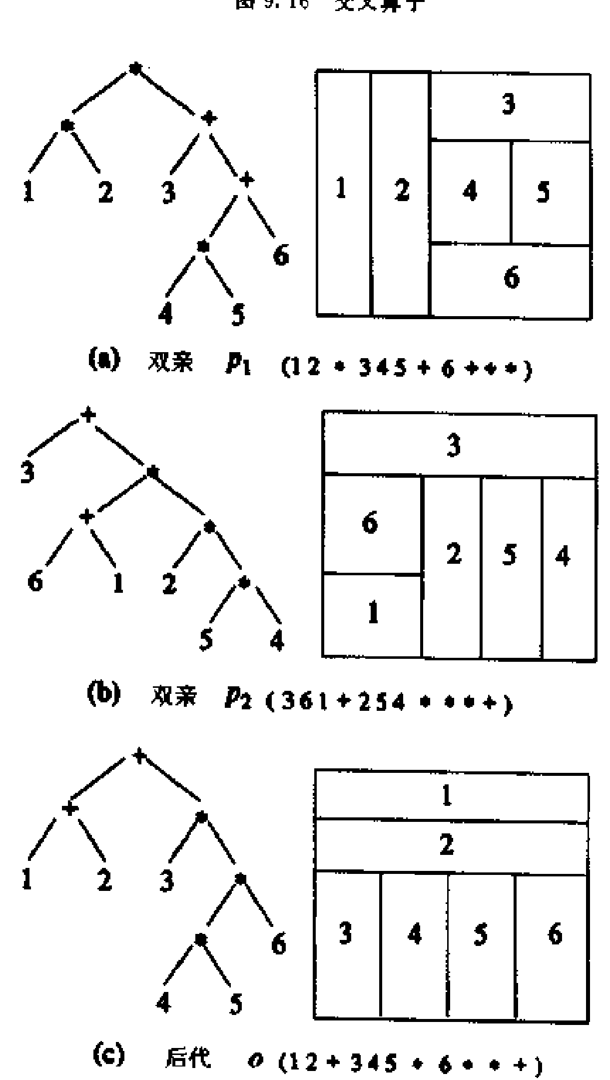


图 9.17 交叉后的布局

9.5.6 变异

随机交换，反转和替代被用作变异操作，交换两个相邻的操作数或两个相邻算子，如图 9.18(a)所示；反转相邻操作数或相邻算子序列，如图 9.18(b)所示；将一个算子替代成相反的，如图 9.18(c)所示。以这种方式进行的变异也可保证产生合法后代。图 9.19 给出了双亲和后代的布局。变异的整个过程如下：

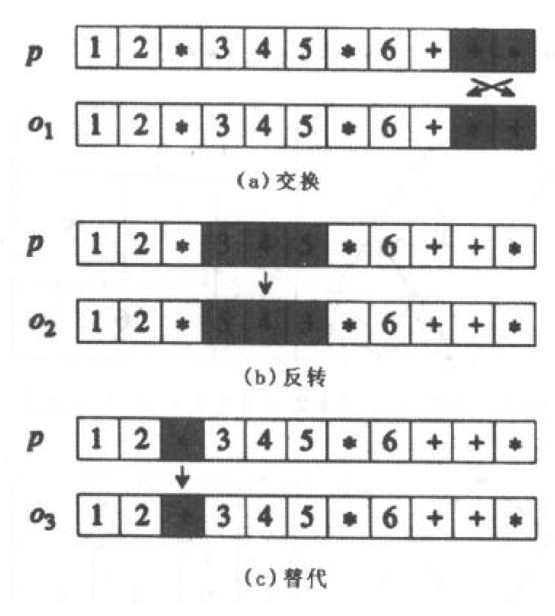


图 9.18 变异算子

变异过程

begin

  $i \leftarrow 0$；

  while $i \leqslant pop_size \times p_m$ **do**

    随机选择一个染色体(未变异的)；

    任选基因；

    if 它是一个算子 **then**

      以相等机会选择交换、反转或替代变异；

    else

      以相等机会选择交换或反转变异

    end

    进行选择的变异；

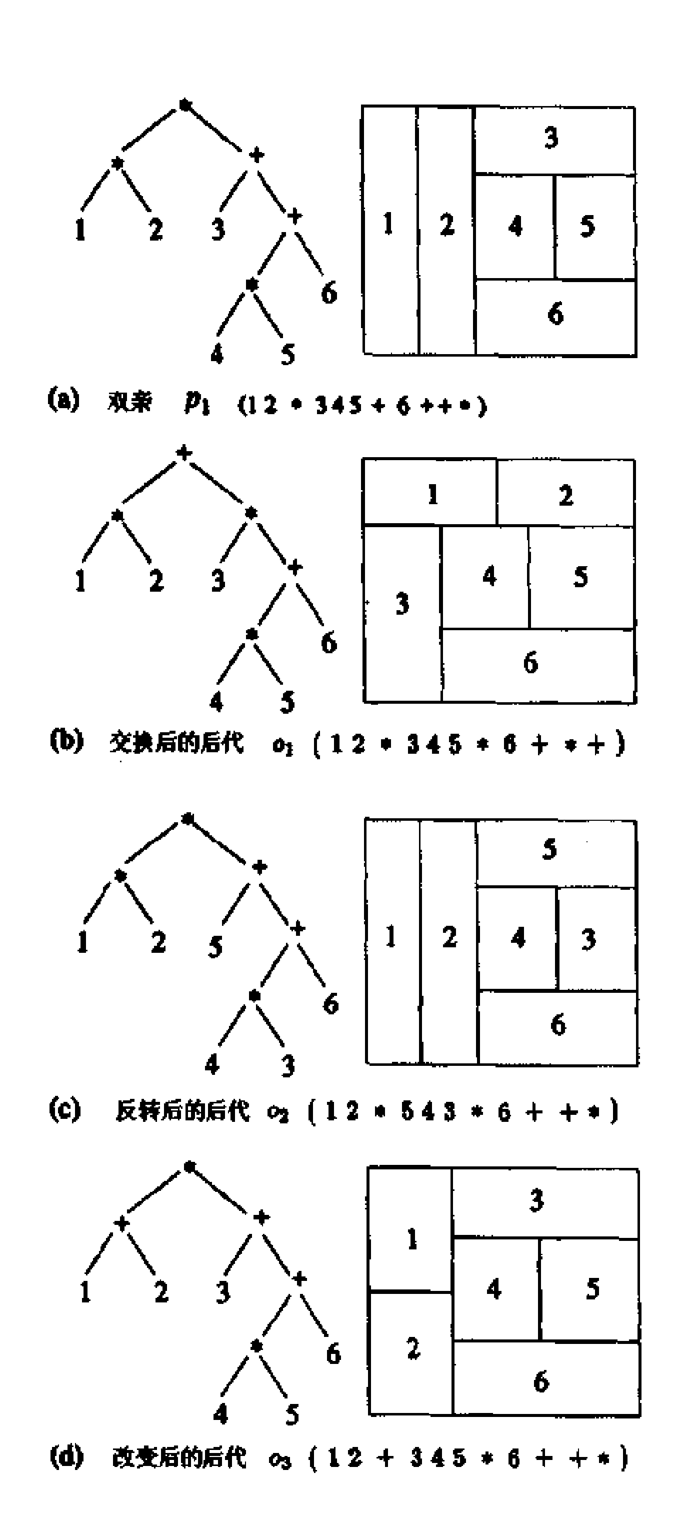
    $i \leftarrow i+1$；

  end

end

9.5.7 从染色体建立一个布局

在评估每个染色体之前，我们必须将它转成分割结构。这以从上至下(top-down)方式进行，从分割树的根部开始，递归地向下通过各个子节点。对于给定的染色体(或分割



(a) 双亲 p_1 (12 * 345 + 6 ++ *)

(b) 交换后的后代 o_1 (12 * 345 * 6 + * +)

(c) 反转后的后代 o_2 (12 * 543 * 6 + + *)

(d) 改变后的后代 o_3 (12 + 345 * 6 + + *)

图 9.19 变异后的布局

树)，首先将其分成两个子树：左子树和右子树。子树也是合法的分割树，对于每个子树，我们再进一步将它分成两个子树。这一过程重复下去，直到每个子树只有一个节点为止。

对于一个染色体，有一个将其分成左子树和右子树的位置，我们称这个位置为分割点(cut-point)。对于一个总尺寸为 $2m-1$ 的分割树 ψ，令 $i(1\leqslant i\leqslant 2m-2)$是树上一个任意分割点(或位置)，令 $n_i(\psi)$表示从分割点右侧到树的最右侧包含的操作数的总数量，$m_i(\psi)$表示树的同一侧包含的算子总数量。在一个染色体上或一个分割树上寻找这一分割点的条件由下述命题给出：

命题 9.2 如果分割点 i 是第一个位置，在 $2m-1$ 到 1 的范围内从右向左查找使等式

$$n_i(\psi) = m_i(\psi) \tag{9.30}$$

成立的位置。则在分割点 i 分割树可以分成①包括给定分割树中从 1 到 i 的元素的左子树；②包括给定分割树中从 $i+1$ 到 $2m-2$ 的元素的右子树。每个子树是一个合法的分割树。

容纳 M_i 的区域 r_i 的可能尺寸 (x_i, y_i) 的集合可以利用区域需求信息在递归分割过程中确定。对于一个区域 r_i，我们用 x_i^l 和 x_i^r 分别表示区域的左边界和右边界，用 y_i^u 和 y_i^b 分别表示区域的上界和下界。区域尺寸和边界的关系计算如下：

$$x_i = x_i^r - x_i^l; \qquad i = 1, 2, \cdots, m \tag{9.31}$$

$$y_i = y_i^u - y_i^b; \qquad i = 1, 2, \cdots, m \tag{9.32}$$

值得注意的是简单树是长度为 3 的特殊的分割树。由染色体建立布局的整个过程如下：

```
建立过程 (树,长度)
begin
  i ← length ;
  while i > 0 do
    将树 tree 分成两个子树;
    计算 x_i^r, x_i^l, y_i^u, y_i^b;
    if 右子树是简单树,then
      计算 x_i^r, x_i^l, y_i^u, y_i^b;
      i ← i−2;
      if 左邻树是简单树,then
        计算 x_i^r, x_i^l, y_i^u, y_i^b;
        i ← i−2;
      else
        construct (左树,长度);
        i ← i−length;
    else
      construct (右树,长度);
      i ← i−length;
      if 左邻树是简单树,then
        计算 x_i^r, x_i^l, y_i^u, y_i^b;
        i ← i−2;
      else
        construct(左树,长度);
        i ← i−length;
      end
    end
  end
  用 x_i^r, x_i^l, y_i^u, y_i^b 计算 x_i 和 y_i;
```

end

图 9.20 是将染色体(1 2 ＊ 3 4 5 ＊ 6 ＋ ＊ ＋)转换为布局的一个实例。

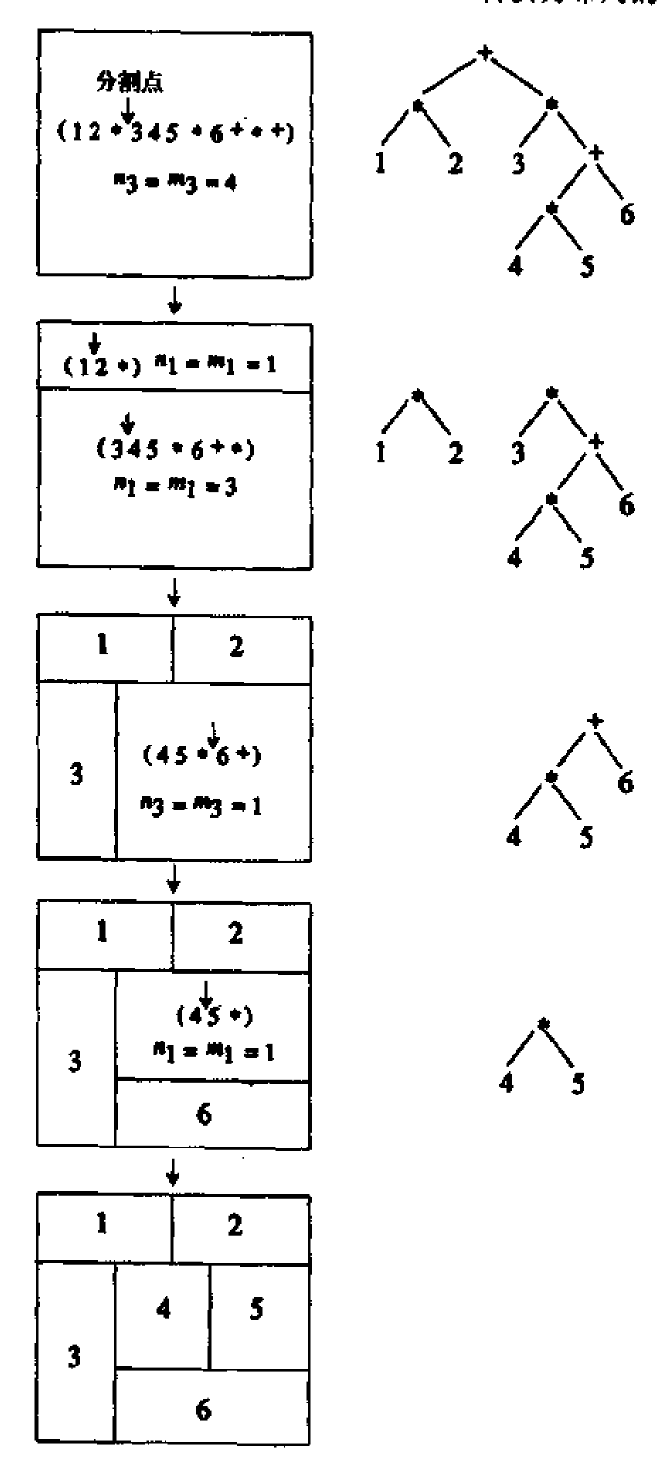


图 9.20 建立布局

9.5.8 评估和选择

在扩大的种群(双亲和后代)中，每个布局由总交通费用目标函数评估，并赋予一个适值。令 v_k 表示扩展种群中第 k 个染色体，$\tilde{C}_k(v_k)$ 为总费用，$d_{ij}(v_k)$ 为每对设备 i 和 j 中心之

间的 Manhattan 距离的实数，$\tilde{c}_{ij}(v_k)$为设备 i 和 j 之间模糊混合的梯形模糊数。对于某一布局，总的评估如下：

$$\widetilde{C}_k(v_k) = \sum_{i=1}^{m}\sum_{j=1}^{m} \tilde{c}_{ij}(v_k)\ d_{ij}(v_k) \tag{9.33}$$

由于总费用是梯形模糊数，不可能直接把它当作适值，因此我们采用 Lee 和 Li 的排序方法计算每个布局总费用的平均值。由于处理的是最小化问题，我们必须将这个平均值转换为适值，以使适合的染色体具有大的适值。转换可以简单地通过对平均值取倒数完成。

由于存在设备外观比率约束，这种方法产生的布局可能存在一个或多个设备违反外观比率约束方面的不合法性，因而我们采用惩罚方法来处理这种不合法性[382]。令 $m(\widetilde{C}_k(v_k))$表示平均值，λ_k 是第 k 个染色体中不符合外观比率约束的总设备数，P 是大数惩罚值。我们在适值函数中加入惩罚项，则有

$$eval(v_k) = \frac{1}{m(\widetilde{C}_k(v_k)) + \lambda_k P} \tag{9.34}$$

其中 $eval(v_k)$是第 k 个染色体的适值函数。

然后，我们用适值建立转轮，繁殖下一代。我们在实验中将精选法嵌入转轮法选择中，以加强对下一代中最好染色体的保护，并克服样本的随机误差。

9.5.9 数值实例

例 9.1 测试问题包括 15 台设备[59]，每个设备的面积和外观比率见表 9.1，用 TrFN 描述的模糊混和见表 9.2。

表 9.1 设备的几何约束

设备标识	面积	外观比率	
		下限	上限
1	100	0.7	1
2	80	1	1
3	50	0.7	1.3
4	60	0.5	0.8
5	120	0.9	1
6	40	0.6	1
7	20	0.7	1.4
8	40	1	1
9	150	0.8	1.1
10	120	0.5	1.5
11	50	0.7	1.1
12	10	0.8	1.2
13	20	0.95	1.5
14	30	0.75	1.25
15	50	0.9	1.1

表 9.2 设备间的模糊混和

	1	2	3	4	5
1	(0,0,0,0)	(8,10,15,18)	(0,0,0,0)	(2,5,8,10)	(0,1,4,12)
2	(8,10,15,18)	(0,0,0,0)	(0,1,2,3)	(2,3,6,8)	(1,2,5,6)
3	(0,0,0,0)	(0,1,2,3)	(0,0,0,0)	(8,10,14,16)	(0,2,5,6)
4	(2,5,8,10)	(2,3,6,8)	(8,10,14,16)	(0,0,0,0)	(0,1,4,6)
5	(0,1,4,12)	(1,2,5,6)	(0,2,5,6)	(0,1,4,6)	(0,0,0,0)
6	(0,0,0,0)	(0,2,4,5)	(0,0,0,0)	(0,1,3,5)	(2,3,5,8)
7	(0,1,2,3)	(0,2,3,6)	(1,2,3,5)	(2,5,6,9)	(4,5,7,9)
8	(1,2,5,10)	(2,3,5,7)	(3,5,6,7)	(0,0,0,0)	(3,5,7,9)
9	(1,2,3,4)	(1,2,3,5)	(2,4,6,8)	(0,0,0,0)	(0,5,8,10)
10	(1,2,4,5)	(0,0,0,0)	(1,5,8,10)	(0,2,4,6)	(0,1,4,6)
11	(0,2,3,6)	(0,2,7,8)	(0,2,4,5)	(0,1,2,3)	(0,0,0,0)
12	(0,0,0,0)	(0,0,0,0)	(1,2,3,6)	(0,0,0,0)	(2,3,6,7)
13	(2,4,5,9)	(6,10,12,15)	(3,5,7,9)	(1,2,5,8)	(0,0,0,0)
14	(0,0,0,0)	(4,5,8,9)	(0,5,8,9)	(2,5,8,9)	(3,5,8,10)
15	(0,0,0,0)	(0,0,0,0)	(3,5,10,12)	(0,0,0,0)	(2,5,9,10)

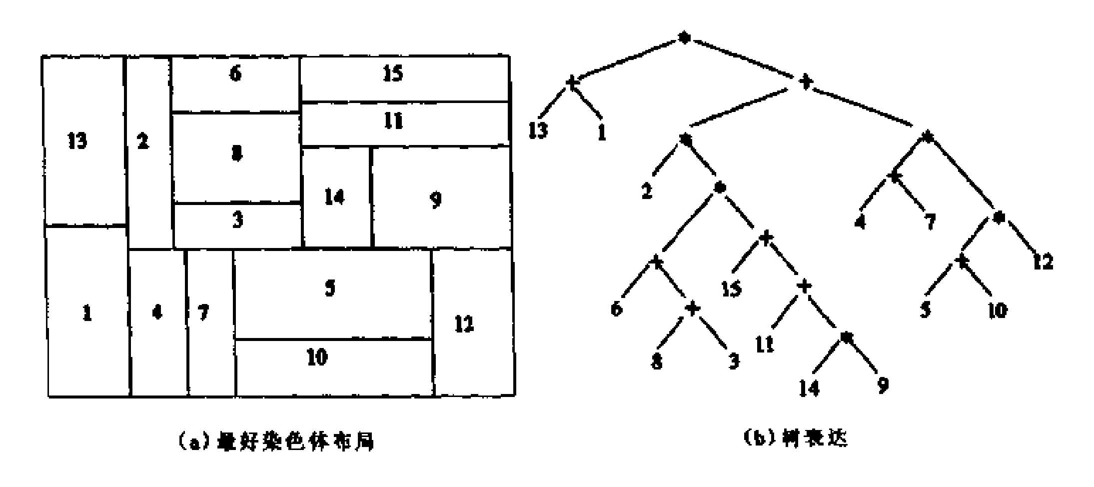
	6	7	8	9	10
1	(0,0,0,0)	(0,1,2,3)	(1,2,5,10)	(1,2,3,4)	(1,2,4,5)
2	(0,2,4,5)	(0,2,3,6)	(2,3,5,7)	(1,2,3,5)	(0,0,0,0)
3	(0,0,0,0)	(1,2,3,5)	(3,5,6,7)	(2,4,6,8)	(1,5,8,10)
4	(0,1,3,5)	(2,5,6,9)	(0,0,0,0)	(0,0,0,0)	(0,2,4,6)
5	(2,3,5,8)	(4,5,7,9)	(3,5,7,9)	(0,5,8,10)	(0,1,4,6)
6	(0,0,0,0)	(1,2,3,4)	(0,2,3,4)	(0,1,2,4)	(4,5,6,8)
7	(1,2,3,4)	(0,0,0,0)	(4,6,8,10)	(0,0,0,0)	(0,1,2,3)
8	(0,2,3,4)	(4,6,8,10)	(0,0,0,0)	(3,5,7,8)	(0,2,4,6)
9	(0,1,2,44)	(0,0,0,0)	(3,5,7,8)	(0,0,0,0)	(0,0,0,0)
10	(4,5,6,8)	(0,1,2,3)	(0,2,4,6)	(0,0,0,0)	(0,0,0,0)
11	(0,0,0,0)	(4,5,8,10)	(7,10,15,18)	(2,10,12,15)	(0,0,0,0)
12	(0,0,0,0)	(3,5,7,9)	(0,0,0,0)	(1,5,8,9)	(2,4,6,9)
13	(1,2,4,5)	(2,5,6,7)	(2,5,9,10)	(5,10,15,18)	(0,0,0,0)
14	(3,5,8,9)	(0,1,3,4)	(0,0,0,0)	(0,0,0,0)	(0,0,0,0)
15	(8,10,12,15)	(0,0,0,0)	(0,0,0,0)	(0,2,7,8)	(4,5,8,10)

	11	12	13	14	15
1	(0,2,3,6)	(0,0,0,0)	(2,4,5,9)	(0,0,0,0)	(0,0,0,0)
2	(0,2,7,8)	(0,0,0,0)	(6,10,12,15)	(4,5,8,9)	(0,0,0,0)
3	(0,2,4,5)	(1,2,3,6)	(3,5,7,9)	(0,5,8,9)	(3,5,10,12)
4	(0,1,2,3)	(0,0,0,0)	(1,2,5,8)	(2,5,8,9)	(0,0,0,0)
5	(0,0,0,0)	(2,3,6,7)	(0,0,0,0)	(3,5,8,10)	(2,5,9,10)
6	(0,0,0,0)	(0,0,0,0)	(1,2,4,5)	(3,2,8,9)	(8,10,12,15)
7	(4,5,8,9)	(3,5,7,9)	(2,5,6,7)	(0,1,3,4)	(0,0,0,0)
8	(7,10,15,18)	(0,0,0,0)	(2,5,9,10)	(0,0,0,0)	(0,0,0,0)
9	(2,10,12,15)	(1,5,8,9)	(5,10,15,18)	(0,0,0,0)	(0,2,7,8)
10	(0,0,0,0)	(2,4,6,9)	(0,0,0,0)	(0,0,0,0)	(4,5,8,10)
11	(0,0,0,0)	(2,5,8,10)	(0,0,0,0)	(3,5,8,12)	(0,0,0,0)
12	(2,5,8,10)	(0,0,0,0)	(2,3,5,6)	(1,3,5,7)	(0,0,0,0)
13	(0,0,0,0)	(2,3,5,6)	(0,0,0,0)	(6,10,14,17)	(0,2,5,8)
14	(3,5,8,12)	(1,3,5,7)	(6,10,14,17)	(0,0,0,0)	(3,4,6,7)
15	(0,0,0,0)	(0,0,0,0)	(0,2,5,8)	(3,4,6,7)	(0,0,0,0)

实验的进化环境如下：*pop_size*=40，p_c=0.4，p_m=0.4，p=5 000，*max_gen*=200。典型解（遗传算法运行一次获得的最好染色体）如下：

[13 1 + 2 6 8 3 + + 15 11 14 9 * + + * * 4 7 * 5 10 + 12 * * + *]

布局见图 9.21(a)，相应的树由图 9.21(b)表达，进化过程见图 9.22。



(a) 最好染色体布局　　(b) 树表达

图 9.21

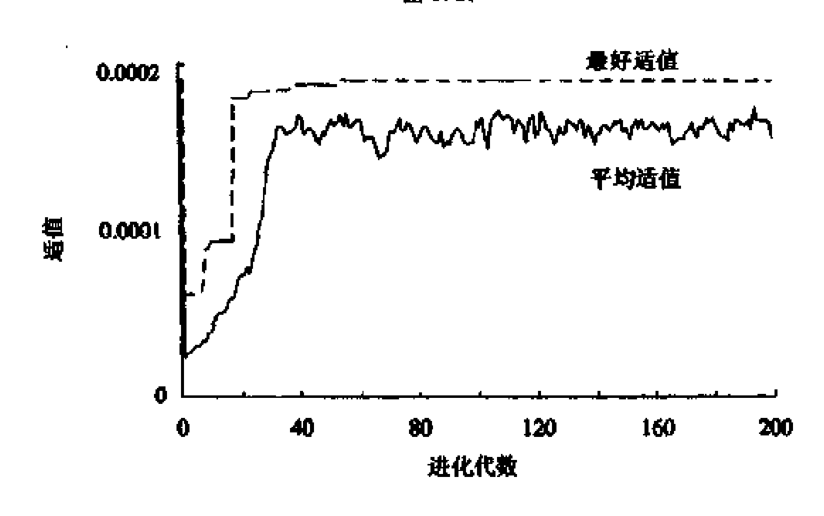


图 9.22　进化过程

从进化过程可知，第 10 代前，所有染色体都具有违反外观比率的不合法性，罚函数将遗传搜索引向可行域。进入可行域后，惩罚失去它的功能，自然选择迫使染色体逐渐进化。在第 63 代，找到最好解。

最好布局的模糊费用指标是 $\tilde{C}$=(2 946.91，5 841.40，9 561.81，12 613.53)。我们可用可能性理论和模糊积分解释模糊现象。模糊数支集的一个元素取确定值的可能性，定义为该元素取确定值的难易程度[429]。许多情况下，可能性就是该成员的隶属函数值。例如，布局的总费用是4 394个单位的可能性是 0.5，如图 9.23 所示。可能性的概念很难在绝对意义上理解，但是，在相对意义上，它是非常有用的。例如，当将一个支撑集合值与另一个支撑集合值对比时，管理者根据它们的可能性，就能够区别出哪个值更可能。遗传算法可以为管理者决策提供一组较好的解。

对管理者可能更有用的是隶属函数密度分布的比例度量，这一度量可通过在所关心区域上的模糊数积分与以支集上的模糊数积分的比值计算：

$$\tilde{I}(x \leqslant \beta) = \frac{\int_{a}^{\beta} \mu(x)dx}{\int_{S} \mu(x)dx} \tag{9.35}$$

例如，由公式(9.35)，4 394 的模糊积分 $\tilde{I}$ 是 0.13。管理者可将这一结果解释为总费用小于或等于 4 394 的布局的模糊期望值是 0.13。9 562 的模糊积分 $\tilde{I}$ 是 0.91，所以总费用小于或等于 9 562 的布局的模糊期望值是 0.91。

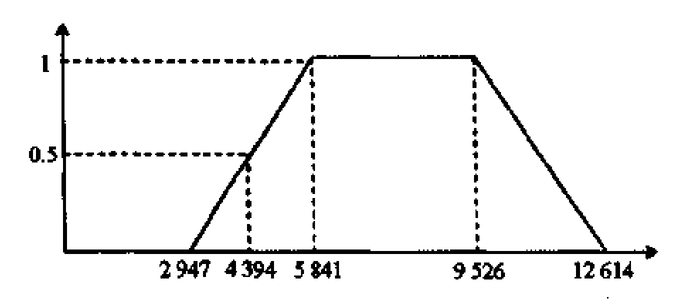


图 9.23 $\tilde{C}$=4 394 的可能性

模糊混和的特性可由四元组(a, b, c, d)描述，其中 a 描述设备间物料流乐观的估计(最好情况)，d 描述设备间物料流悲观的估计(最差情况)，b 描述设备间物料流平均的估计(近于最好情况)，c 描述设备间物料流另一平均的估计(近于最差情况)。根据四种情况，我们可以用同样的模糊数据建立四个等价的非模糊问题，并用所提出的算法求解(稍作修改使它适用于非模糊情况)。四个问题的每次运行都采用同样的参数设置，每种情况下最好解如下：

最好情况：[5 14 3 * 9 * * 15 10 + 12 13 * 1 * * 6 7 8 11 + * * 4 * 2 * + +]

近于最好：[6 7 + 8 * 9 1 2 + * * 5 15 11 13 * 10 3 * 14 * 4 12 * * + * * +]

近于最差：[9 15 * 14 * 4 13 1 * * * 10 12 6 * + 11 3 8 7 + + 5 * 2 * * * +]

最差情况：[7 3 * 14 * 4 * 11 6 * + 2 * 12 * 9 13 8 * * + 5 * 10 15 1 + + *]

图 9.24 描述了最好情况的布局和相应的树描述；图 9.25 是近于最好情况的布局和相应的树描述；图 9.26 是近于最差情况的布局和相应的树描述，图 9.27 是最差情况的布局和相应的树描述。

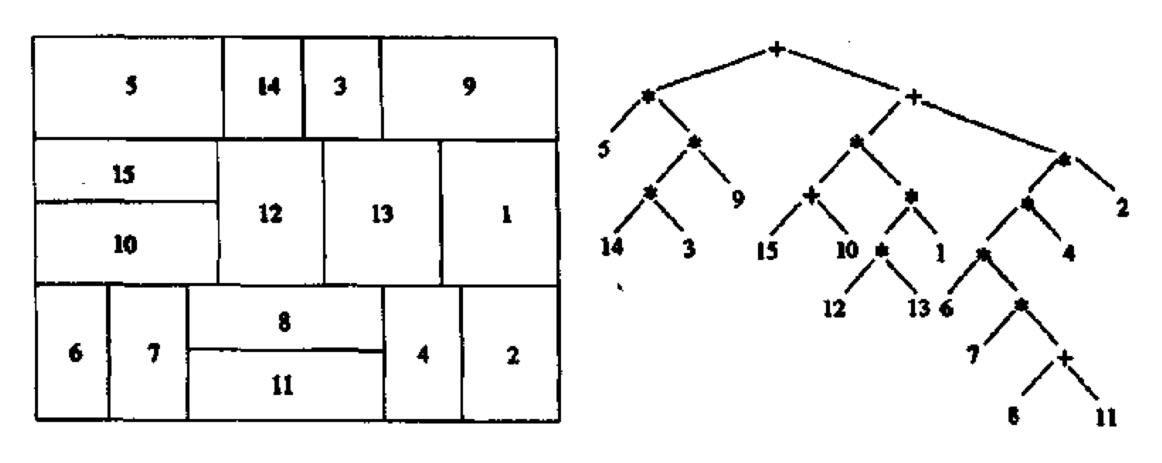


图 9.24 最好情况的布局和树描述

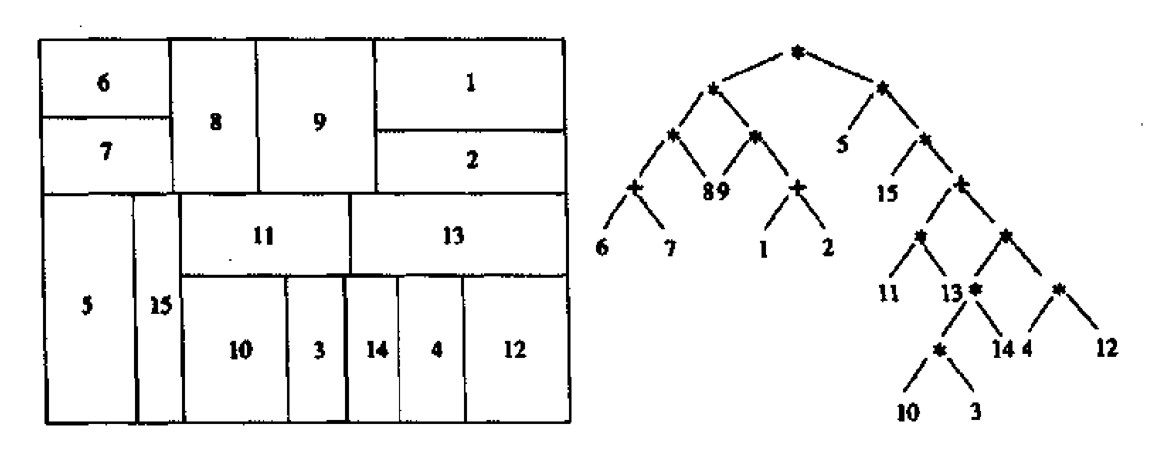


图 9.25 近于最好情况的布局和树描述

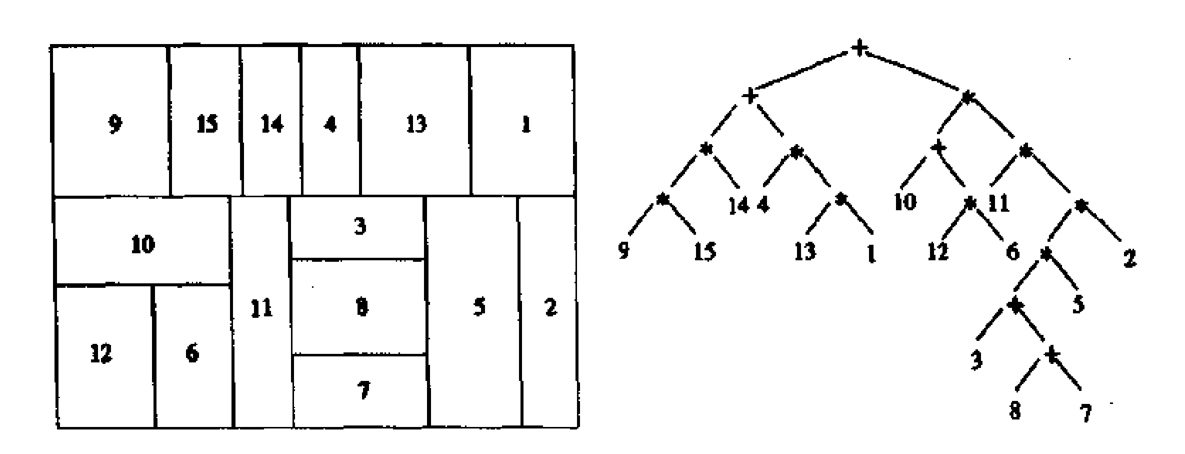


图 9.26 近于最差情况的布局和树描述

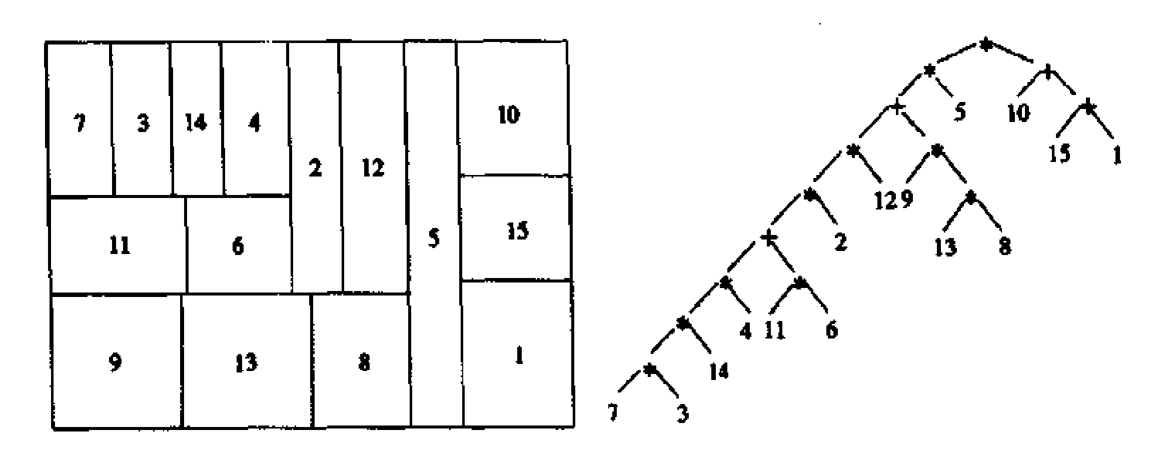


图 9.27 最差情况的布局和树描述

对于每种布局，我们计算四种物流情况（最好、近于最好、近于最差、最差）的总费用，并与模糊解对比，见表 9.3。

表 9.3 结果对比

解的类型	a	b	c	d
模糊	2 947	5 841	9 562	12 614
最好情况	2 795	6 068	9 953	13 215
近于最好情况	2 895	5 083	9 625	12 922
近于最差情况	2 971	5 869	9 581	12 685
最差情况	3 012	6 257	10 024	13 134

由对比结果可知，在考虑物流时，一个非模糊方法在其等价的非模糊情况下，可获得最小费用布局，而在其他情况下，这一方法获得的布局的费用比用模糊方法获得的布局的费用要高。即非模糊方法只能得到适合其考虑情况的合理布局，而模糊方法可获得适合从最好到最差所有情况的合理解。例如，对于物流最好情况，等价非模糊问题的总费用是2 794，它比模糊情况总费用2 947小。采用同一布局，其他三种情况（近于最好情况、近于最差情况、最差情况）的费用分别是6 068，9 953，13 215，它们比模糊情况的值要大。用模糊方法得到的解的相对误差在图 9.28 中给出。

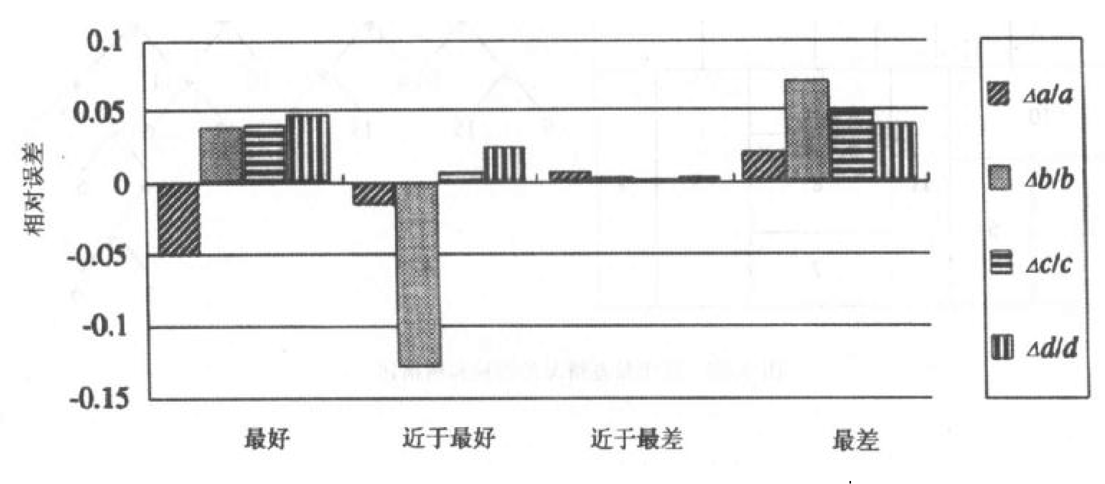


图 9.28 模糊解的相对误差

第十章　工程设计中的选择专题

10.1　资源受限项目调度问题

由于 Kelley 的早期工作，许多研究者提出了资源受限项目的调度问题[245]。通常，调度决策同时服从于时序约束和资源约束。对此许多研究者处理了不同情况：一个或两个约束是松驰的，或至少是简化的。因为资源受限项目调度问题必须处理时序（前后接续关系）约束和资源约束，所以，它比一般调度问题更难解决[21，386]。

通常的资源受限项目调度问题可描述为：问题具有一系列相互关联的活动，（有时序关系），其中，每一活动可以以几种模式（方式）完成，每一模式以已知延续时间和给定资源需求为特征。这一问题的解是在满足时序约束和资源约束的条件下产生一种使某些管理目标为最优的调度，即每个活动何时开始和采用哪种资源/延续时间的模式[42]。

早期的研究是采用数学规划中的标准解决技术寻求问题的准确最优解，如整数规划[75，341，388]、有界枚举（bounded enumeration）[97]、分枝定界[333]和隐枚举[394]。由 Blazewicz 给出的结果[40]，容易看出资源受限项目调度问题是 NP 难题。对于大的项目，问题的规模将使得优化计算方法难以实施。在这种情况下，应采用启发式解决方法，即采用能够产生合理次优调度的相当简单的调度规则。在过去的 30 年中，开发和测试了大量的启发式算法，Alvarez-Valdés 给出了这方面的综述和计算对比[7]。

现在已知的许多启发式方法可看作是优先派遣规则（priority dispatching rules），这一规则在根据时序相关的启发规则或资源相关的启发规则制定解决资源冲突的排序决策时给出活动的优先级。Patterson 等提出了回溯算法（backtracking algorithm），这一算法考虑了各种目标函数形式和各种模式下的活动，具有很强的柔性[334]。Bell 和 Han 开发了两阶段启发式算法，这一算法的性能比其他启发式的性能都好[31]。Sampson 和 Weiss 用局域搜索技术解决了一般资源受限项目的调度问题，结果表明，局域搜索启发式优于已有的启发技术[365]。Jeffcoat 和 Bulfin 介绍了他们用模拟退火解决在考虑准备时间和完工期限情况下的资源受限调度问题的工作[235]。Drexl 和 Gruenewald 研究了非预占（nonpreemptive）多模式的（multi-mode）资源受限项目调度问题，并提出了一种随机调度方法[113]。

10.1.1　问题描述

我们考虑一般的资源受限项目调度问题：

（1）一个单一项目包括许多延续时间已知的活动；

（2）每个活动的开始时间依赖于其他一些活动的完成（时序约束）；

（3）在调度期内，资源消耗为常量；

（4）资源在每个时段上是有限的，但时段之间是可更新的；

（5）资源之间不可替换；

(6) 活动不可被中断；

(7) 每个活动只有一种执行模式；

(8) 管理目标是最小化项目延续时间。

问题的数学描述如下：

$$\min \quad t_n \tag{10.1}$$

$$\text{s.t.} \quad t_j - t_i \geqslant d_i, \quad \forall j \in S_i \tag{10.2}$$

$$\sum_{t_i \in A_{t_i}} r_{ik} \leqslant b_k; \quad k = 1,2,\cdots,m \tag{10.3}$$

$$t_i \geqslant 0; \quad i = 1,2,\cdots,n \tag{10.4}$$

其中：

t_i——活动 i 的开始时间；

d_i——活动 i 的延续时间(处理时间)；

S_i——活动 i 的紧后活动集合；

r_{ik}——活动 i 对资源 k 的需求量；

b_k——资源 k 的总可用量；

A_{t_i}——在时间 t_i 处理的活动的集合；

m——不同资源类型的数量。

活动 1 和 n 是标识项目开始和结束的虚(dummy)活动，目标是极小化项目总延续时间。约束(10.2)是时序约束，约束(10.3)是资源约束，以确保在任何时段，所有活动使用资源 k 的量不超过资源 k 的限制量。根据问题，我们可以定义时序图 $G=(V,H)$。H 为时序约束集合，V 为项目活动集合。G 是有向非循环的。一个简单的网络描述如图 10.1 所示。

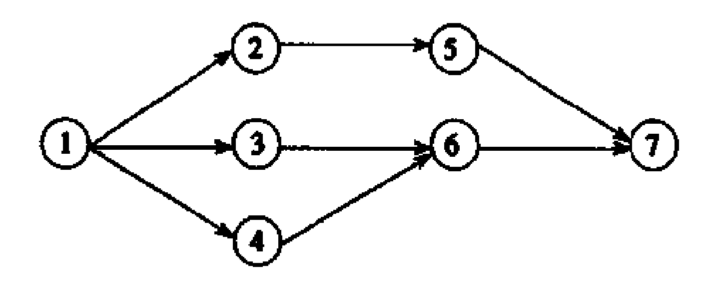


图 10.1 项目的网络描述

10.1.2 混和遗传算法

Cheng 和 Gen 提出了一种资源受限项目调度问题的混和遗传算法[62,64]。这里，遗传算法用作亚策略(meta-strategy)，而将局域搜索技术和特定于问题的启发式算法与之结合以增进其性能。对于这一问题，有两个重要的问题需要解决：

(1) 在不影响时序约束的情况下确定活动的顺序；

(2) 根据可用资源确定最早开始时间。

该算法的基本思路是：用遗传算法寻找活动的合理顺序，用特定于问题的启发式算法确定最早开始时间。

1. 表达

关于调度问题已提出许多表达机制。在某种意义上，所有这些表达可看作是顺序表达的扩展[71]。

通常，关于调度问题所提出的表达机制可分为直接表达和间接表达[15]。以前几乎所有关于调度问题的 GA 研究采用的都是间接描述，即遗传算法工作在编码解的种群之上，并且从染色体表达向合法调度转换必须在评估之前由调度建立者完成。例如 Nakano 的二进制表达[313]、Syswerda 的顺序列表表达[392]、Bagchi 的顺序/处理、计划/资源列表表达[15]和 Davis 的时序列表表达[100]。

在直接问题表达方式中，调度本身用作染色体，因而，没有解码过程；但是，需要开发复杂的遗传算子。例如 Kanet 的顺序/机器/开始时间列表表达[241]和 Bruns 的完全调度表达[50]。

资源受限项目调度问题本质上可看作是一类服从某些约束的排序问题，因为一旦活动的顺序给定了，每个活动的最早可能开始时间就可根据可用资源很容易地确定。我们关心的是如何用遗传算法找到一个活动的合适顺序。在非直接表达中，我们用排好顺序的活动列表对问题进行染色体编码。这样，染色体表达活动可能的排列。令 $\mathbf{v}_k$ 表达当前种群中第 k 个染色体，令 i_j 是染色体在 j 位置上的活动，则染色体可表达如下：

$$\mathbf{v}_k = [i_1, i_2, \cdots, i_j, \cdots, i_n]$$

遗传算法的搜索空间是活动所有可能排列的空间，但由于时序约束的限制，可行空间只是整个空间的一部分。

2. 初始化

正如 Davis 和 Steenstrup 所指出的，初始化过程可通过随机产生的种群或选择适用种群来完成[102]。对于资源受限项目调度问题，由于存在时序约束，采用随机产生种群方法进行初始化将会产生大量的不可行调度。

一种一次通过启发过程被用于种群的初始化。它以固定顺序一次考虑一个仍未调度的活动，调度是从左向右添满的。在每一阶段，保持一个调度的集合，并完全随机地给出冲突活动的优先级。这一过程一直重复下去，直到所有活动被调度。它可以保证产生时序可行的调度。令 $\mathbf{v}_k^t$ 是 $\mathbf{v}_k$ 的一个部分染色体，包括前 t 个活动。Q_t 是在阶段 t 与给定的 $\mathbf{v}_k^t$ 对应的可调度活动的集合，令 P_i 是活动 i 所有紧前活动的集合，则与给定 $\mathbf{v}_k^t$ 对应的可调度活动集合 Q_t 定义为

$$Q_t = \{j \mid P_j \subset \mathbf{v}_k^t\}$$

它包括尾节点在特定调度 $\mathbf{v}_k^t$ 中的活动，并且是所有竞争活动的集合。产生初始调度过程如下：

产生初始调度的过程

```
begin
  k ← 1;
  while k ≤ pop_size do
    t ← 1;
```

```
      v_k ← {1};
      while t ≤ n do
        修改可调度活动集合 Q_t;
        从 Q_t 中任选活动 j;
        v_k ← v_k ∪ {j};
        t ← t+1;
      end
      k ← k+1;
    end
  end
```

3. 评估

通常，评估包括两个主要步骤：

(1) 评估原始目标值；

(2) 将目标函数值转为适值。

如我们所知，问题的目标是最小化项目延续时间。首先，我们必须计算所有染色体活动的最早可能开始时间，由于每个染色体对应于特定的可能活动顺序，因此给定染色体的开始时间可根据给定活动顺序和可用资源从左向右计算。令 σ_i 为活动 i 最早可能开始时间，ϕ_i 为活动 i 最早可能结束时间，d_i 为活动 i 的延续时间，b_{kt}为阶段 t 时资源 k 的当前可用量。对于一个给定的染色体 $\mathbf{v}_k$ 和一个在任意位置的活动 j，最早可能开始时间 σ_j 可确定如下：

$$\sigma_j = \max\ \{t \mid t > \sigma_{j\min}\text{且}$$
$$b_{kt}, b_{k(t+1)}, b_{k(t+2)}, \cdots, b_{k(t+d_j)} \geqslant r_{jk};\ k = 1,2,\cdots,m\ \}$$

其中

$$\sigma_{j\min} = \max\ \{\phi_i \mid i \in P_j\}$$

于是最早可能完成时间 ϕ_j 为 $\sigma_j + d_j$。

由于处理的是最小化问题，我们必须将原始目标函数转换为适值函数，以确保适合的个体具有大的适值，这可由下述转换过程完成：

$$g(\mathbf{v}_k) = \frac{f_{\max} - f(\mathbf{v}_k)}{f_{\max} - f_{\min}}$$

其中 $g(\mathbf{v}_k)$为适值函数。$f(\mathbf{v}_k)$是原始目标值，即项目延续时间。$\mathbf{v}_k$ 是当前代的第 k 个染色体。$f_{\max}$和 $f_{\min}$是当前代原始目标函数的最大值和最小值。

将 γ 增量转换用于适值函数：

$$g(\mathbf{v}_k) = \frac{f_{\max} - f(\mathbf{v}_k) + \gamma}{f_{\max} - f_{\min} + \gamma} \tag{10.5}$$

其中 γ 是开区间(0,1)上的一个正实数。采用这一转换的原因有两个：①避免等式(10.5)被零除；②使选择活动从适值-比例选择向完全随机选择的调整成为可能。

4. 遗传算子

交叉和变异是遗传算法的两个主要遗传算子。通常，在实现中交叉作为主要的遗传算子，而变异只作为辅助方法，以在搜索空间引入最小的可能改变。如我们所知，理想的遗传搜索应能在搜索空间的扩展(exploitation)和探索(exploration)间达到平衡[105]。Cheng 和 Gen 采用一种新的方法控制遗传算子的设计。两个算子的理想实现是：一个设计成广泛搜索，以尽量扩展超越局域最优的区域；另一个设计成集中搜索，以尽量利用可改善的最好解。如果我们设计集中遗传算子，我们必须用领域特定知识增强这一算子。对于许多组合优化问题，任意遗传算子(交叉或变异)可根据问题性质采用集中策略设计.采用这种方法，变异和交叉同样重要。在 Cheng 和 Gen 的实现中，用交叉进行广泛搜索，变异进行集中搜索。

(1) 交叉

对于排序或排列的描述，人们研究了许多重组合算子，由于活动间存在时序约束，这些交叉算子不能直接用于这一问题。这里所提出的交叉算子包括两个主要步骤。首先，进行通常的 PMX (部分映射交叉法)运算[174]，对活动重新排序；然后，进行修复过程，解决时序冲突。由于 PMX 算子可能搅乱活动的时序关系，因而有必要解决时序冲突以保证后代是时序可行调度。交叉算子运行机制如下：

交叉过程

begin

  $i \leftarrow 0$;

  while $i \leqslant pop_size \times p_c$ **do**

    任选两个个体；

    用通常的 PMX 方法产生一个后代；

    解决后代的时序冲突；

    $i \leftarrow i+1$;

  end

end

令 S_j 表示活动 j 所有紧后活动的集合，对于一个给定的后代 $\mathbf{v}_k = [i_1, i_2, \cdots, i_j, \cdots, i_n]$，令 $\mathbf{v}_k^j$ 表示包括前 j 个活动的部分调度，即 $\mathbf{v}_k^j = [i_1, i_2, \cdots, i_j]$ $(j \leqslant n)$。修复过程如下：

修复过程

begin

  $j \leftarrow 2$;

  while $j \leqslant n-1$ **do**

    $j \leftarrow j+1$;

    检查 $\mathbf{v}_k^{j-1}$ 中的每个元素，判断它是否属于 S_j；

    if 存在 m 个元素属于 S_j **then**

      将它们放在$[j+1, n-1]$之间的任意位置；

      $j \leftarrow j-m$;

    end

```
    end
  end
```

（2）变异

变异算子采用集中搜索策略设计，结合邻域技术寻找改进的后代。将调度 x 不超过 λ 个基因的变化所得到的调度集合称为调度 x 的领域。在这种意义上，如果 x 比其邻域的任何其他解好，则调度 x 被称作 λ 优化。变异算子运行机制如下：

变异过程

begin

  $i \leftarrow 0$;

  while $i \leqslant pop_size \times p_m$ **do**

    任选一个没有进行变异的染色体；

    在染色体中任取 λ 个基因的子调度；

    通过两两互换产生邻域调度；

    对于不合理邻域，解决时序冲突；

    评估所有邻域调度；

    选择最好邻域作为后代；

    $i \leftarrow i+1$;

  end

end

我们看一个实例。假设子调度长度为 $\lambda = 3$，任选(3 4 5)。子调度中基因的可能排列和染色体中的不变基因(假设所有的排列都是合理的)形成邻域染色体，如图 10.2 所示。



图 10.2 变异：邻域染色体

5. 选择

采用转轮法(符合-比例选择方法的一种)进行选择，精选法(elitist way)嵌入其中，以加强对下一代中最好染色体的保护并克服样本的随机误差。采用精选法选择，如果上一代最好个体没有繁殖新的一代，则需从新一代中任意去掉一个个体，并将上一代中的最好个体加入下一代。

10.1.3 实例

考虑文献[96,98]中的两个标准实例问题。上述算法用 C 语言编程，在 NEC EWS 4800/260 工作站上运行。

例 10.1 文献[98]中给出的问题共有 27 个活动，其中包括两个虚活动；每个活动对三种不同资源有固定的需求，需求量是单位量的整数倍。可用资源在项目延续时间上是常量。问题如图 10.3 所示。

资源限制： b_1=6 个单位，b_2=6 个单位，b_3=6 个单位

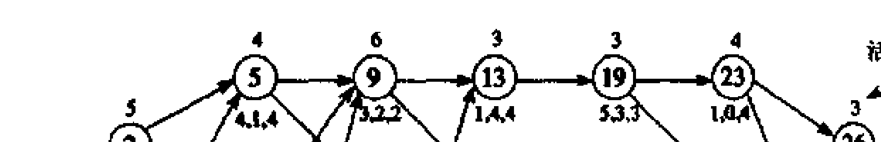
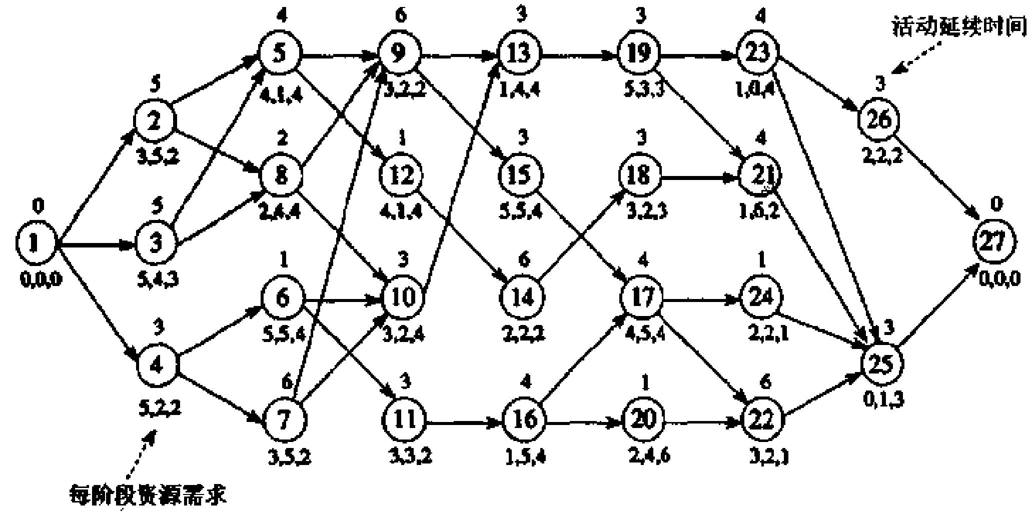



图 10.3 第一个实例(由 Davis 和 Patterson 给出)[98]

这一实例的进化环境是：$pop_size=40$，$p_c=0.3$，$p_m=0.3$。最好染色体在第 12 代产生：

* 目标函数 *

f^*：64

* 最好调度 *

v^* = [1 2 4 3 7 8 6 10 11 5 16 20 12 9 14 15 17 13 24 22 18 19 23 21 25 26 27]

* 开始时间 *

[0 0 5 8 13 15 21 25 26 27 27 33 36 36 39 43 47 48 48 49 52 55 57 57 61 61 64]

* 完成时间 *

[0 5 8 13 15 21 25 26 27 33 33 36 39 39 43 47 48 49 54 52 55 57 61 61 64 64 64]

图 10.4 给出了与其他启发式方法结果的对比。这些启发式方法是最小最迟完成时间(Minimum Late Finish Time，LFT)，最大资源利用率(Greatest Resource Utilization，GRU)，最短邻域操作(Shortest Imminent Operation，SIO)，最小工作松弛时间(Minimum Job Slack，MINSLK)，资源调度法(Resource Scheduling Method，RSM)，任选工作法(Select Jobs Randomly，RAN)，最多工作可能(Most Jobs Possible，MJP)和最大资源需求(Greatest Resource Demand，GRD)。这些结果表明，这里提出的算法优于所有已有的启发式算法。

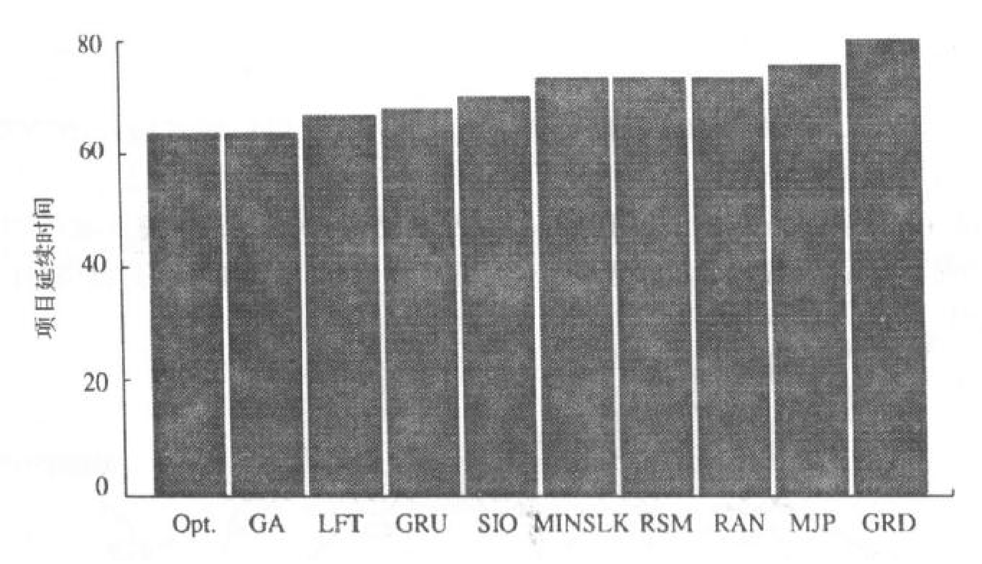


图 10.4 与其他启发式结果对比

图 10.5 给出了第一个实例的进化过程。由这些结果可知，这里提出的算法可迅速找到已知最优。

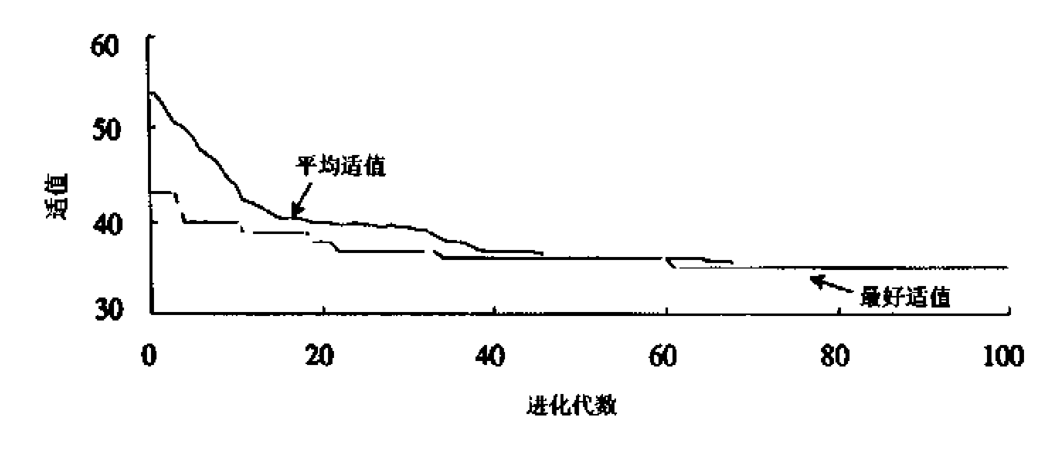


图 10.5 第一个实例的进化过程

例 10.2 文献[96]中给出的实例有 27 个活动，其中包括两个虚活动；每个活动对三种不同资源有固定的需求，需求量是单位量的整数倍，资源可用量在项目延续时间上是常量。问题如图 10.6 所示。

第二个实例的进化环境与第一个实例相同。最好染色体在第 29 代产生：

* 目标函数 *

f^*：35

* 最好调度 *

v^* = [1 3 2 4 7 8 5 12 6 10 11 14 9 15 16 13 18 19 20 24 23 17 22 21 25 26 27]

* 开始时间 *

[0 0 0 3 4 6 6 10 11 14 14 18 18 18 20 21 22 22 24 24 27 27 27 30 30 32 35]

* 完成时间 *

[0 4 3 6 10 11 10 18 14 16 18 20 20 21 22 24 24 27 26 27 30 30 32 32 33 35 35]

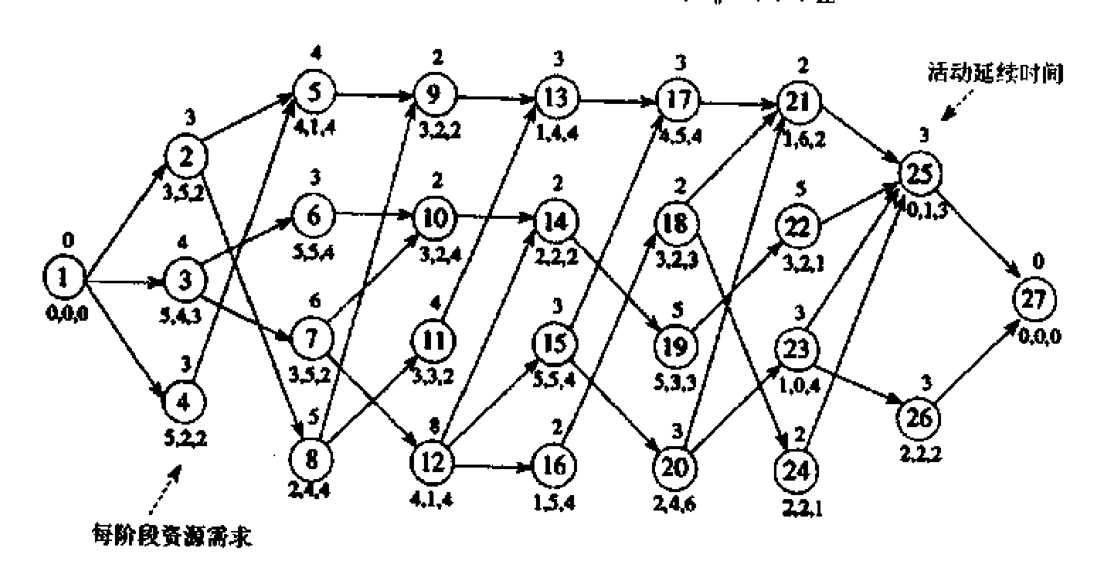


图 10.6　第二个实例[96]

图 10.7 给出了第二个实例的进化过程。100 次运行的解分布如图 10.8 所示。

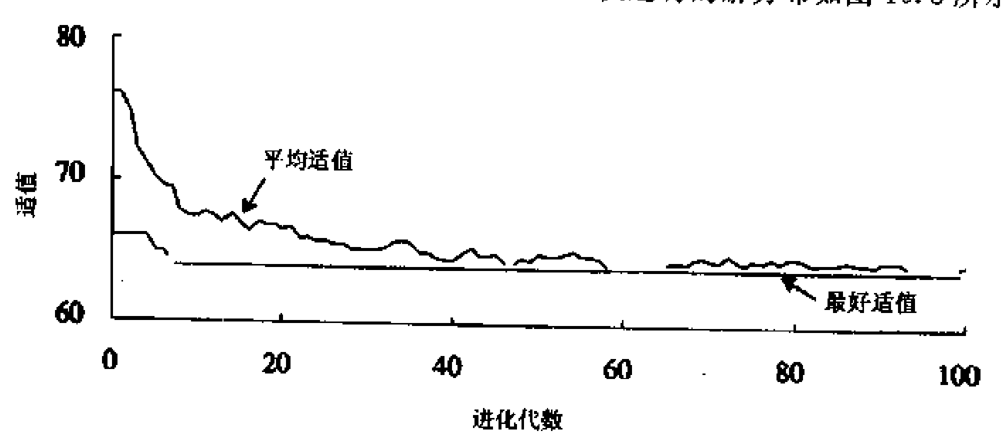


图 10.7　第二个实例的进化过程

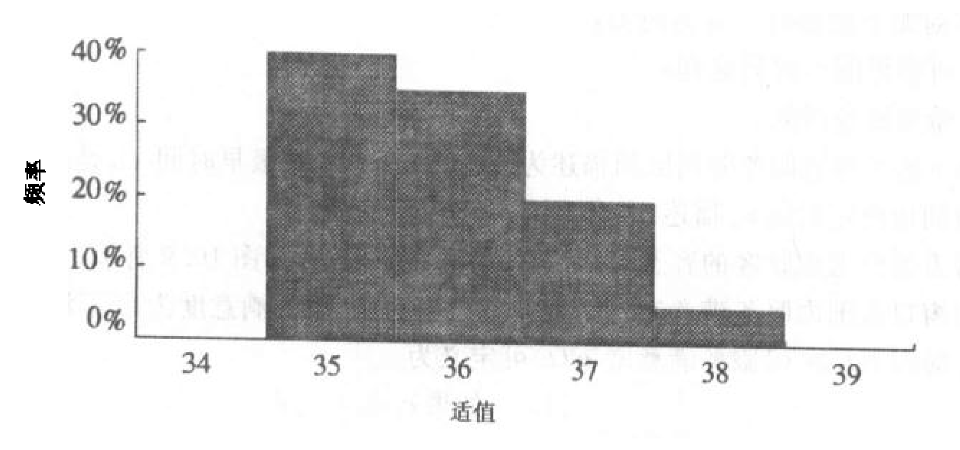


图 10.8　100 次运行的解分布

10.2 模糊车辆路径与调度问题

车辆路径问题(Vehicle Routing Problem,VRP)是指为服务于需求已知的一组顾客的一个车队,设计一组开始和结束于一个中心出发点的最小费用路径。每个顾客只被服务一次,而且,一个车辆服务的顾客数不能超过它的能力[43]。

具有时间窗口约束的车辆路径问题(Vehicle Routing Problem with Time Window Constraints,VRPTW)是车辆路径问题 VRP 的一类重要的扩展。在 VRPTW 中,每个顾客有一个时间窗口(由截止时间和最早时间约束组成的时间区间),在这个区间内必须对他服务。当时间窗口存在时,问题包括路径组合和调度因素。路径设计必须使总运输费用最小,同时,调度必须保证时间可行。

在许多实际应用中,时间窗口概念没有很好地表达出顾客的偏好。尽管要求顾客提供固定的服务时间窗口,顾客却希望尽可能在希望的时间得到服务,我们称这一希望的时间为约定时间(due-time)。在这种情况下,顾客的偏好信息可表达为满足服务时间意义上的凸模糊数,Cheng 和 Gen 提出用模糊约定时间替换时间窗口的概念。因为它可以比固定时间窗口更好地满足顾客偏好。当在模糊服务时间上进行调度时,我们感兴趣的不仅是通常对于所有顾客可行的服务时间,而且是在努力使服务时间尽可能接近每个顾客的约定时间意义上的合理服务时间。Cheng 和 Gen 描述了模糊车辆路径问题(Fuzzy Vehicle Routing Problem,FVRP),并提出了一种解决它的遗传算法[65,69]。

10.2.1 问题描述

模糊车辆路径问题是基于模糊约定时间的概念描述的,其中模糊约定时间的隶属函数符合服务时间的满意度。这里考虑的目标是最小化车队大小、最大化平均顾客满意度、最小化总行进距离和总等待时间[63]。

1. 模糊约定时间

顾客对服务的偏好可分为两类:

(1) 可容忍服务时间区间;

(2) 希望服务时间。

顾客 i 的可容忍服务时间区间描述为$[e_i, l_i]$,其中 e_i 是最早时间,l_i 是最迟时间。希望服务时间由约定时间 u_i 描述,通常 $e_i \leqslant u_i \leqslant l_i$。

通常方法只考虑顾客的容忍时间,并用时间窗口表达,如图 10.9 所示。如果服务时间落在时间窗口范围内服务满意度为 1[完全满意];否则,服务满意度为 0(不满意)。即对于任意服务时间 $t(t > 0)$服务满意度 $\mu_i(t)$可定义为

$$\mu_i(t) = \begin{cases} 1, & \text{如果 } e_i \leqslant t \leqslant l_i \\ 0, & \text{其他} \end{cases} \tag{10.6}$$

如我们所知,如果我们试图同时处理两类顾客偏好,自然应将顾客偏好描述为考虑服务时间满意度的三角形模糊数,定义为三元组(e_i, u_i, l_i)。例如,如果顾客在希望时间被服务,他的满意度为 1 (完全满意);否则,满意度随着服务时间和希望时间之间的不同的增

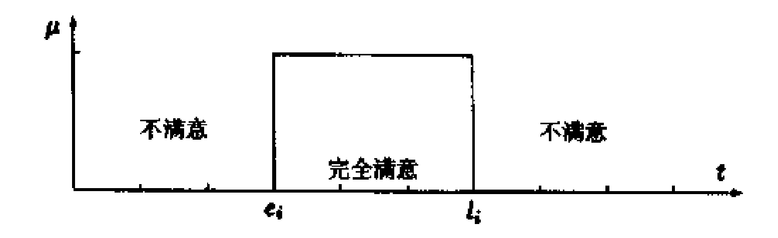


图 10.9 时间窗口

大而减小。如果服务时间落在时间区间外，满意度为0(不满意)。我们称这样的三角函数为顾客 i 的模糊约定时间，如图 10.10 所示。

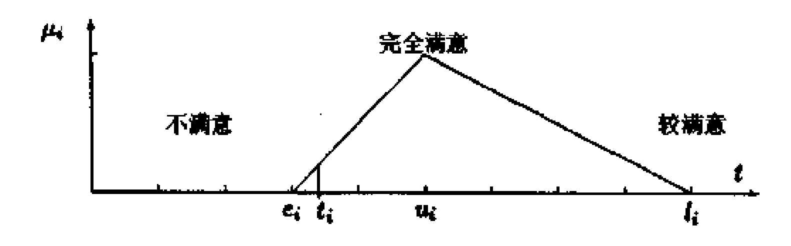
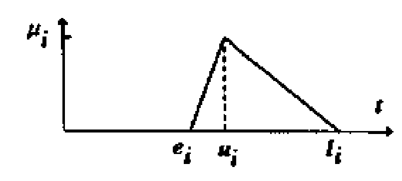


图 10.10 模糊约定时间

令顾客 i 模糊约定时间的隶属函数为 $\mu_i(t_i)$，它代表服务时间为 t_i 时的满意度，隶属函数定义为

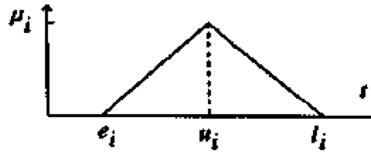
$$\mu_i(t_i)=\begin{cases}0, & t_i<e_i\\ \dfrac{t_i-e_i}{u_i-e_i}, & e_i\leqslant t_i\leqslant u_i\\ \dfrac{l_i-t_i}{l_i-u_i}, & u_i\leqslant t_i\leqslant l_i\\ 0, & t_i>l_i\end{cases}\tag{10.7}$$

图 10.11 表示具有同样希望服务时间但对提前和拖后有不同容忍度的四种类型顾客偏好，即

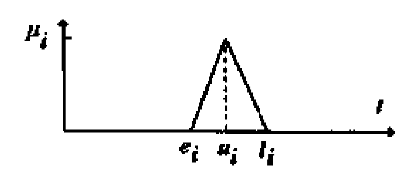


(a)对拖期容忍度比对提前容忍度大

(b)对提前容忍度比对拖期容忍度大



(c)对提前和拖期都有一定容忍度



(d)对提前和拖期都有较小容忍度

图 10.11 典型顾客偏好

(1) 对拖期容忍度比对提前容忍度大；

(2) 对提前容忍度比对拖期容忍度大；

(3) 对提前和拖期都有一定容忍度；

(4) 对提前和拖期都有较小容忍度。

如果车辆 k 从顾客 i 直接驶向顾客 j 并很早到达顾客 j，它将等待。车辆等待时间由下述等式确定：

$$w_j = t_j - (t_i + r_{ij}) \tag{10.8}$$

式中，r_{ij}是从顾客 i 到顾客 j 的行进时间；w_j 是车辆在顾客 j 的等待时间。他们的关系如图 10.12 所示。

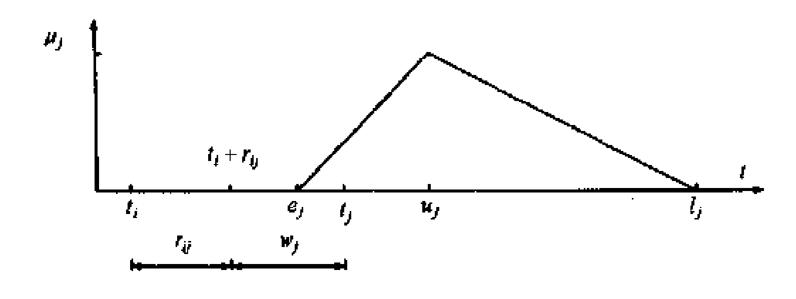


图 10.12 等待时间

2. 模型

为准确描述这一问题，我们首先介绍一些概念，然后给出一个多目标优化描述。

(1) 常量

n——顾客数；

m——车辆数；

c_k——车辆 k 的能力；

a_i——给顾客 i 的递送量；

d_{ij}——从顾客 i 直接到顾客 j 的行进距离；

$w_i(t_i)$——当服务时间为 t_i 时，车辆在顾客 i 的等待时间；

$\mu_i(t_i)$——顾客 i 的满意度。

(2) 变量

t_i——顾客 i 的服务时间；

$$y_{ik} = \begin{cases} 1, & \text{如果顾客 } i \text{ 由车辆 } k \text{ 服务} \\ 0, & \text{其他} \end{cases}$$

$$x_{ijk} = \begin{cases} 1, & \text{如果车辆 } k \text{ 直接从顾客 } i \text{ 驶向顾客 } j \\ 0, & \text{其他} \end{cases}$$

则 F_VRP 模型可描述如下：

(3) 模糊车辆路径问题描述

$$\min \quad \sum_{j=1}^{n} \sum_{k=1}^{m} x_{0jk} \tag{10.9}$$

$$\max \quad \frac{1}{n} \sum_{i=1}^{n} \mu_i(t_i) \tag{10.10}$$

$$\min \quad \sum_{k=1}^{m}\sum_{i=0}^{n}\sum_{j=0}^{n} d_{ij}\, x_{ijk} \tag{10.11}$$

$$\min \quad \sum_{i=1}^{n} w_i(t_i) \tag{10.12}$$

$$\text{s.t.} \quad \mu_i(t_i) > 0, \quad \forall\, i \tag{10.13}$$

$$\sum_{i=1}^{n} a_i\, y_{ik} \leqslant c_k, \quad \forall\, k \tag{10.14}$$

$$\sum_{k=1}^{m} y_{ik} = 1, \quad \forall\, i \tag{10.15}$$

$$\sum_{i=0}^{n} x_{ijk} = y_{jk}; \quad \forall\, j,\, k \tag{10.16}$$

$$\sum_{j=0}^{n} x_{ijk} = y_{ik}; \quad \forall\, i,\, k \tag{10.17}$$

$$\sum_{ij \in S\times S} x_{ijk} \leqslant |S| - 1, \quad \forall\, k, \quad S \subseteq \{1, 2, \cdots, n\} \tag{10.18}$$

$$x_{ijk} = 0 \text{ 或 } 1; \quad \forall\, i,j,k \tag{10.19}$$

$$y_{ik} = 0 \text{ 或 } 1; \quad \forall\, i,k \tag{10.20}$$

$$t_i \geqslant 0, \quad \forall\, i \tag{10.21}$$

式中下标 0 表示车队中心点。目标(10.9)表示最小化车队大小;目标(10.10)是最大化平均顾客满意度;目标(10.11)是最小化总运行距离;目标(10.12)最小化总等待时间。约束(10.13)确保对每个顾客的服务时间在时间容忍区间内;约束(10.14)确保每台车辆搭乘的顾客不超过它的能力;约束(10.15)确保每个顾客只由一台车辆服务;约束(10.16)和(10.17)对于每一个顾客,只有两个顾客与之相连,车辆由一个直接驶向它,又由它直接驶向另一个;约束(10.18)描述车辆 k 直接运行和顾客数的关系。容易看出,这一描述中存在两个已知组合优化问题。约束(10.14)和(10.15)是一般指派问题的约束。如果变量 y_{ik} 一定满足约束(10.13)~(10.15),则对于给定 k,约束(10.16)~(10.18)定义了一个对顾客分配车辆 k 的旅行商问题。

通常考虑的目标:

(1) 最小化车队大小;

(2) 最小化总行进距离。

通常重点放在最小化车队大小目标上,但必须在车队大小和行进距离间进行权衡。

对于模糊车辆路径问题,除了这些目标,还必须考虑另外两个目标,即

(1) 最大化平均顾客满意度;

(2) 最小化对车辆的总等待时间。

重点放在顾客平均满意度目标上,这将导致车辆的总等待时间的增加。因此,我们也需要在顾客平均满意度和车辆总等待时间之间进行权衡。

当在非模糊情况下进行决策时,通常方法只能保证每个顾客的服务时间在允许的时间窗口内,不能考虑顾客的希望服务时间。在考虑顾客偏好的模糊性时,我们不仅像通常方法那样对所有顾客服务时间的可行性感兴趣,而且,在尽量对每个顾客的服务时间达到

他的约定时间的意义上，对服务时间的合理性感兴趣，以达到最大化平均满意度的目的。

10.2.2 相关的遗传算法研究

过去几年中，一些学者介绍了他们关于用遗传算法解决具有时间和能力约束的车辆路径问题的研究。Thangiah，Vinayagamoorthy 和 Gubbi 研究了具有时间期限(deadline)的车辆路径问题[400]，目标是在不超过车辆能力和不拖期情况下，最小化服务于一组顾客的车辆数量和行进距离。他们提出了一种聚类第一路径第二(cluster-first route-second)的启发式过程。遗传算法用于对顾客分类(给顾客分配车辆)，然后，用最廉价插入(cheapest insertion)启发式为每类顾客(或每个车辆)寻找路径。用这一启发式方法得到的解，是对标准 2-优算法的改进，以确定是否对某些顾客应由相邻群中的车辆服务。

Blanton 和 Wainwright 研究了具有时间窗口和能力约束的车辆路径问题[39]，目标是确定在符合车辆能力约束和顾客允许时间窗口约束条件下，为所有顾客服务所需的最小车辆数。他们试图给出一种一步全局优化过程，而不是像 Thangiah 等提出的分阶段过程，顾客排列表用作编码机制，并提出了基于最近邻域启发式的交叉算子。在评估阶段，Davis 的贪婪启发式用于将顾客排列表解码成车辆调度。贪婪启发式首先取初始化 k 台车辆的排列表中的前 k 个顾客，然后对剩余的 $n-k$ 个顾客分别检查。每选择一个新顾客，就对每台车辆评估可能的子巡回，并选择最好的子巡回。他们的遗传算法实现有两个问题，一个问题是解码过程将毁坏染色体中的子巡回，因而进化在此失去意义。另一个问题是解码过程主要依赖于前 k 个顾客，因为交叉算子的实质是一类最近邻域启发式，前 k 个顾客不适于初始化 k 台车辆。

Potvin 和 Dube 将遗传算法用来为车辆路径启发式算法搜索好的参数设置，并证明这种方法通过在参数空间上的遗传搜索可以在很大程度上改进并行插入启发式的结果[339]。

10.2.3 混和遗传算法

Cheng 和 Gen 提出了一种解决模糊车辆路径与调度问题的混和遗传算法[65,69]。他们设计了一种基于插入启发式(insertion heuristic-based)的交叉算子用于寻找改良的后代。为了处理问题的模糊特性，提出了一种推-碰撞-掷(push-bump-throw)过程。通过这一过程，模糊车辆路径问题在交叉运算中可由两个主要步骤完成。

(1) 首先把问题只看作是具有时间窗口的一般问题，并用顺续插入启发式获得可行解。

(2) 然后，将重点放在模糊特性上，用推-碰撞-掷过程确定每个顾客的最好服务时间。

1. *表达方式*

将顾客/服务时间/车辆列表用于染色体表达，它表达了与服务时间和指定车辆有关的顾客排列，其基因是一个顺序的三元组(顾客、服务时间、车辆)，这一表达属于直接表达方式，如图 10.13 所示。

顾客 i_1 服务时间 t_{i_1} 车辆 v_{i_1}	顾客 i_2 服务时间 t_{i_2} 车辆 v_{i_2}	…… …… ……	顾客 i_n 服务时间 t_{i_n} 车辆 v_{i_n}

图 10.13 染色体表达机制

2. 初始化

对于模糊车辆路径问题，由于存在允许服务时间约束，不能用简单的随机过程产生初始种群，这里提出的初始化过程包括三个主要步骤：

(1) 随机产生顾客排列；

(2) 用从左向右扫描过程将顾客对应于车辆进行聚类；

(3) 通过推-碰撞-掷过程确定每个顾客最好服务时间。

初始化过程

```
begin
  i←0;
  while i ≤ pop_size do
    产生顾客排列;
    对应于车辆将顾客聚类;
    确定顾客的最好服务时间;
    i←i+1;
  end
end
```

3. 聚类过程

聚类过程是一个连续地向车辆增加顾客的过程。当向当前车辆增加新的顾客时，必须检查当前车辆的能力可行性和顾客的服务时间可行性。如果两者都可行，新顾客就分给当前车辆，否则，将这个顾客分给新的车辆；然后，计算每个顾客最早可能的服务时间。重复上述过程，直到所有顾客都分给一台车辆为止。整个过程如下：

聚类过程

```
begin 对应一个染色体
  将最左侧顾客分给一台车辆;
  i←1;
  while i ≤ n do
    检查能力、服务时间的可行性;
    if 两者均可行 then
        向当前车辆增加顾客;
      else
        找到一台新车辆;
```

```
      end
      确定顾客最早服务时间；
      i←i+1；
   end
end
```

假设第 k 台车辆有 $i-1$ 个顾客，表示为 $U_k=\{1,\cdots,i-1\}$，考虑是否能将下一个顾客 i 分配给这台车辆。我们需要用下述不等式检查当前车辆能力的可行性：

$$\sum_{j\in U_k} a_j + a_i \leqslant c_k$$

并用下述不等式检查顾客服务时间可行性。

$$t_{i-1} + r_{i-1,i} \leqslant l_i$$

如果两者都可行，顾客 i 分配给当前车辆，顾客 i 最早可能服务时间计算如下：

$$t_i = \begin{cases} e_i, & \text{若 } t_{i-1} + r_{i-1,i} \leqslant e_i \\ t_{i-1} + r_{i-1,i}, & \text{其他} \end{cases}$$

如果能力约束或服务时间不可行，顾客 i 分配给一个新的车辆。

4. 推-碰撞-挪过程

完成上述步骤后，我们得到了一组染色体，每个染色体代表能力和服务时间可行的调度，将每个顾客的服务时间给定为他的最早可能服务时间。我们需要进一步确定每个顾客的最好服务时间以最大化总满意度。这里提出了确定每个顾客最好服务时间的推-碰撞-挪过程，这一过程的基本思想很简单：我们努力使顾客服务时间尽可能靠近他的希望时间（约定时间）。

通常，一个可行染色体包括几个车辆路径计划，而一个车辆的路径计划则包括一些紧路径和/或松路径。

紧路径：任何相邻顾客间具有零等待时间顺序的顾客列表。

松路径：任何相邻顾客间具有非零等待时间顺序的顾客列表。

图 10.14 是紧路径和松路径的示意图。

推-碰撞-挪过程首先从左向右扫描可行调度，试图在紧路径上找到可能的向前推动。这一推动将在不影响时间约束的情况下，增加这条路径上的总满意度。令 P 是表示紧路径顺序的顾客列表，w 表示紧路径后的第一个非零等待时间。顾客 i 的可能前推可确定如下：

$$\Delta_i = \begin{cases} u_i - t_i, & \text{若 } e_i < t_i < u_i \\ l_i - t_i, & \text{若 } u_i < t_i < l_i \end{cases} \tag{10.22}$$

紧路径的可能前推可确定如下：

$$\Delta_{min} = \min\ \{ \min_{i\in P}\{ \Delta_i\},\ w \} \tag{10.23}$$

如果存在这种前推，我们将相关的部分前推。每次前推后，应该进行两种检查：

(1) 前后关系检查：检查 w 是否变为零，如果等待时间变为零，这意味着下一个紧路径与当前紧路径相连。换句话说，当前紧路径碰撞下一个紧路径。

(2) 不可推检查：在已经前推的部分中检查不可推路径，然后，将它们分离出来。因

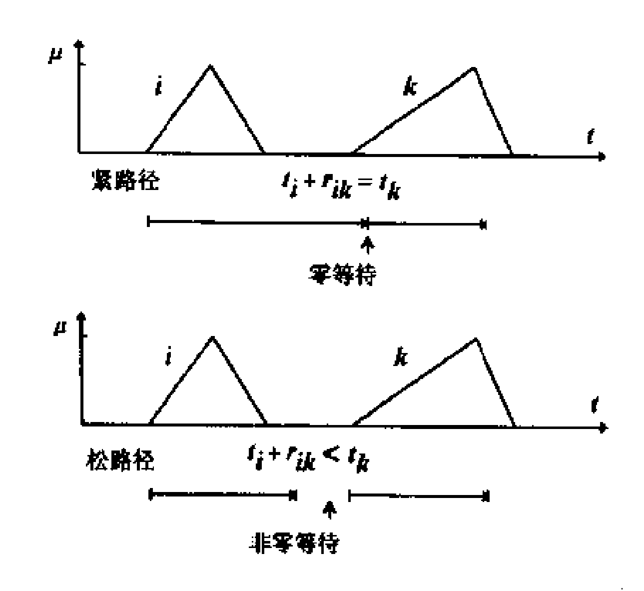


图 10.14 紧路径和松路径示意图

为对不可推路径的任何前推将导致①违反时间约束或②降低总满意度。换句话说，我们从当前紧路径中扔掉(掷)不可推路径。

令 $\mu_i'(t_i)$表示顾客 i 在时间 t_i 满意度函数的斜率，我们将所有在紧路径上的顾客的该斜率求和：

$$\mu' = \sum_{i\in P} \mu_i'(t_i) \tag{10.24}$$

如果 μ' 大于零，一个可能的前推将带来总满意度的增加；否则，这一前推将导致总满意度下降。根据 μ'，我们可以决定是否进行前推，并检查不可前推路径，将它分离出来。在推-碰撞-掷过程中，由于前推、紧路径和松路径会经常发生变化。图 10.15 给出了推-碰撞-掷过程的简单例子。

从图中我们知，第一条紧路径包括顾客 i，k 和 j。这条紧路径的可能前推是 $\Delta_{\min}=\Delta_k$。前推 Δ_k 使顾客 k 的满意度为 1。在顾客 i 和 k 形成不可推路径。扔掉顾客 i 和 k 后，余下路径的可能前推是 $\Delta_{\min}=w$。将余下路径前推 w，顾客 j 与顾客 l 碰撞。

令 P 是当前处理的紧路径，P_k 由 P 的前 k 个顾客组成，P' 是 P 后的紧路径。整个过程如下：

推-碰撞-掷过程

begin

　将最左侧顾客分给一台车辆；

　$P\leftarrow$ 第一条紧路径；

　while 所有顾客处理完毕 **do**

　　不可推检查：

　　if $\sum_{i\in P_k} \mu'_i \leqslant 0$ **then**

　　　$P\leftarrow P\setminus P_k$(扔掉 P_k)；

　　end

　　　为空检查：

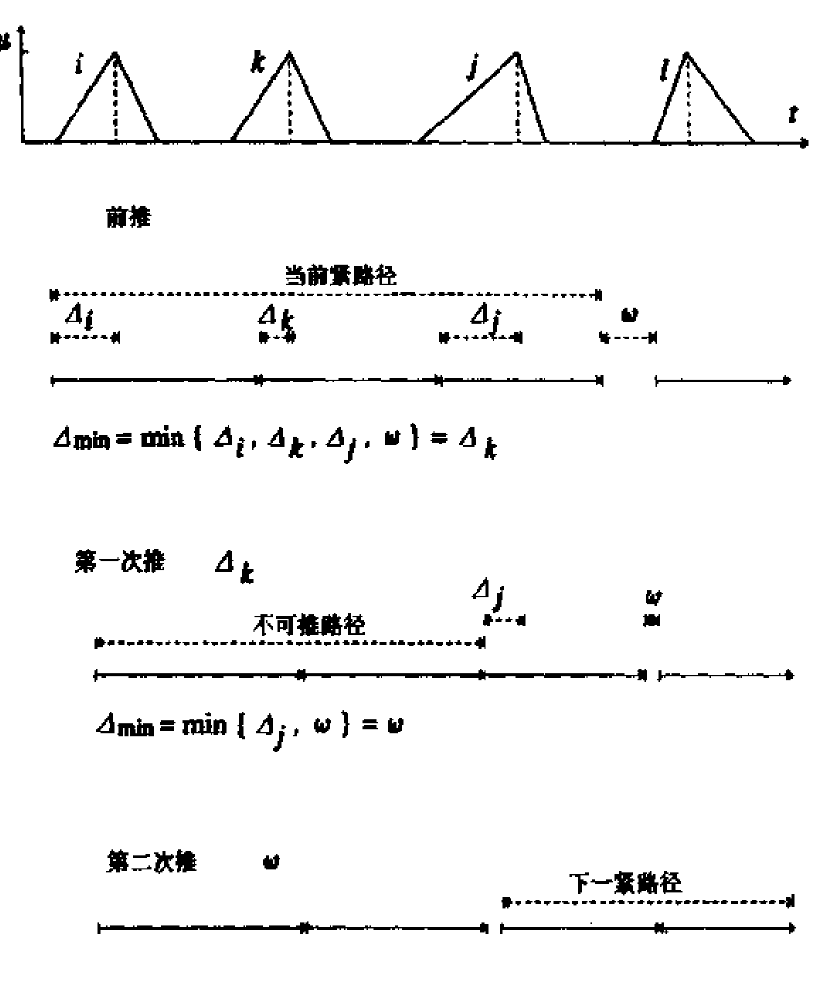


图 10.15 推-碰撞-绑过程示意图

```
if P 为空 then
  P←P';
end
为 P 找出 Δmin;
前推 Δmin;
前后关系检查:
if w = 0 then
  P←P∩P' (扩展 P);
end
end
end
```

5. 遗传算子

具有时间窗口约束的车辆路径与调度问题的启发式算法可分为巡回建立启发式(tour-building heuristics)和巡回改进启发式(tour-improvement heuristics)。连续插入启发式是巡回建立启发式的一种,它首先用"种子"顾客初始化路径,未进入路径的顾客在满

足调度时间和(或)能力约束的条件下,一个个地加入这一路径,直到路径装满。如果还有未进入路径的顾客,则重复初始化和插入过程,直到所有顾客都被服务为止。连续插入算法在一系列路径测试问题上优于其他所有算法[384]。

(1) 可行插入

现在,我们研究可行插入的充分必要条件。假设已有部分建造的可行路径($i_1, i_2, \cdots, i_m$),我们要在顾客 i_{p-1}和 i_p 之间插入一个顾客 k,$1 \leqslant p \leqslant m$。令 $t_{i_p}^{NEW}$是在插入 k 后顾客 i_p 的新的服务时间。如果假设运行距离符合三角不等式,这一插入在调度中定义了一个前推:

$$\Delta t_{i_p} = t_{i_p}^{NEW} - t_{i_p} \tag{10.25}$$

$$\Delta t_{i_r} = \max \{0, \Delta t_{i_{r-1}} - w_r\}, \quad p < r \leqslant m \tag{10.26}$$

如果 $\Delta t_{i_p} > 0$,一些顾客 i_r, $p \leqslant r \leqslant m$ 可能成为时间不可行的。容易看出,我们应该连续地检查这些顾客的时间可行性,直到找到一些用户 i_r,$r < m$,满足 $\Delta t_{i_r} = 0$ 或 i_r 是时间不可行的。最坏的情况是所有顾客 i_r, $p \leqslant r \leqslant m$ 均被检查。我们已证明:

引理 1 在部分建立的可行路径($i_1, i_2, \cdots, i_m$)上,在 i_{p-1}和 i_p($1 \leqslant p \leqslant m$)之间插入一个顾客 k 时,时间可行的充分必要条件是:

$$t_k \leqslant l_k \text{ 和 } t_{i_r} + \Delta t_{i_r} \leqslant l_{i_r}, \quad p \leqslant r \leqslant m$$

(2) 交叉算子

所提出的交叉算子主要基于连续插入启发式,它由两步组成:

1) 用连续插入启发式产生一个能力和服务时间可行的调度;

2) 用推-碰撞-掷过程确定每个顾客最好服务时间。

所提出的过程运行如下:

步骤 1:首先从两个双亲最右侧顾客中任选一个顾客作为后代初始顾客.

步骤 2:适当地交换相关的顾客,以使选择的顾客在两个双亲最后的位置.

步骤 3:将选中的顾客从双亲中移走,将它分给车辆 1 ,并像上节介绍的那样确定它的最早可能服务时间.

步骤 4:现在与选中顾客相邻的两个顾客成为插入的候选顾客,计算他们在后代部分路径中最好的插入位置,并确定最好的插入顾客。

步骤 5:将最好的顾客插入当前车辆,并确定他的最早可能服务时间。

步骤 6:如果两个候选顾客都不能插入当前车辆,将他们中的任意一个放在后代的最后位置,开始一个新的车辆。

步骤 7:重复上述步骤:计算、比较、交换、移动和插入,直到产生一个完整的后代。

步骤 8:产生一个后代后,用推-碰撞-掷过程确定每个顾客最好服务时间。

我们来看一个例子。为了说明简单,我们只用排序顾客列表作为染色体表达方式。考虑两个双亲染色体:

p_1: 5 1 3 2 4 6 7 9 8

p_2: 8 1 2 9 3 5 4 6 7

假设顾客 7 选作后代初始顾客,将他分配给车辆 1,并确定他的最早可能服务时间。现在我们有下述部分后代:

p_1: 5 1 3 2 4 6 8 9 ×

p_2: 8 1 2 9 3 5 4 6 ×

o: 7 × × × × × × × ×

几步后，我们得到下述部分后代：

p_1: 5 1 3 2 8 6 × × ×

p_2: 8 1 2 6 3 5 × × ×

o: 4 9 7 × × × × × ×

两个候选插入顾客为 5 和 6，计算他们的插入费用，并选择一个费用小的（如 5）插入。然后，我们需要交换第一个双亲中顾客 5 和 6 的位置：

p_1: 6 1 3 2 8 5 × × ×

p_2: 8 1 2 6 3 5 × × ×

o: 4 9 7 × × × × × ×

假设在部分后代中最好的插入位置是在顾客 4 和 9 之间，从两个双亲染色体中移走顾客 5，并将其插入部分后代：

p_1: 6 1 3 2 8 × × × ×

p_2: 8 1 2 6 3 × × × ×

o: 4 5 9 7 × × × × ×

将顾客 5 分配给与顾客 4 和 9 相同的车辆，并确定他的最早可能服务时间。重复这些步骤，直到产生完整的后代。

最好插入顾客由下述等式确定：

$$c(i,k,j) = \alpha_1\left(1 - \frac{d_{ij}}{d_{ik} + d_{kj}}\right) + \alpha_2 \frac{w_k^{\text{NEW}} + w_j^{\text{NEW}}}{w_j^{\text{OLD}}} \tag{10.27}$$

$$\alpha_1 + \alpha_2 = 1,\ \alpha_1 \geqslant 0,\ \alpha_2 \geqslant 0 \tag{10.28}$$

其中 w_k^{NEW} 和 w_j^{NEW} 是插入顾客 i 后的等待时间。w_j^{OLD} 是插入顾客 i 前的等待时间。插入指标函数 $c(i,k,j)$ 是伪正常迂路(detour)（第一部分）和迂路节省的伪正常等待时间（第二部分）的加权和。最好插入顾客是使函数取最小值的顾客。

6. 评估和选择

适值根据原始目标函数计算，一个染色体的目标加权和如下：

$$g(\mathbf{v}_k) = \rho_1 \frac{u_k}{u_{\max}^0} + \rho_2\left(1 - \frac{1}{n}\sum_i \mu_{ki}(t_{ki})\right) + \rho_3 \frac{1}{n}\sum_i \frac{w_{ki}}{w_{\max}^0} + \rho_4 \frac{d_k}{d_{\max}^0}$$

$$\sum_{i=1}^{4} \rho_i = 1;\quad \rho_i \geqslant 0;\quad i = 1, 2, 3, 4 \tag{10.29}$$

其中，$\mathbf{v}_k$ 是第 k 个染色体；w_{ki} 是在染色体 $\mathbf{v}_k$ 中顾客 i 的等待时间；$w_{\max}^0$ 是初始染色体中的最大等待时间；u_k 是染色体 $\mathbf{v}_k$ 中车辆总数；$u_{\max}^0$ 是初始染色体中车辆最大数；d_k 是染色体 $\mathbf{v}_k$ 的运行总距离；$d_{\max}^0$ 是初始染色体中最大运行距离。于是一个染色体的适值计算如下：

$$eval(\mathbf{v}_k) = \frac{g(\mathbf{v}_{\max}) - g(\mathbf{v}_k)}{g(\mathbf{v}_{\max}) - g(\mathbf{v}_{\min})} \tag{10.30}$$

用转轮法作选择机制，并用精选法嵌入其中，以保护下一代中的最好染色体并克服采样的随机误差。

10.2.4 实验结果

计算实验是为了验证提出过程的有效性。测试问题包括30个顾客，每个顾客的模糊约定时间随机产生，评估函数和插入费用函数的基本权重的设置见表10.1。遗传算法基本参数设置见表10.2。

表10.1 函数权重的基本设置

ρ_1	ρ_2	ρ_3	ρ_4	α_1	α_2
0.5	0.3	0.1	0.1	0.5	0.5

表10.2 遗传算法参数基本设置

p_c	*pop_size*	*max_gen*
0.3	30	400

基于上述基本设置，交叉概率在0.1到0.5间变动。表10.3给出了每个设置下10次运行结果的均值。从结果可知，交叉概率高，可产生较好的解。

表10.3 不同 p_c 的结果

p_c	μ	w	d	u
0.1	0.825	14.71	1 659	3
0.2	0.841	15.41	1 541	3
0.3	0.897	15.31	1 603	3
0.4	0.883	16.39	1 437	3
0.5	0.907	16.89	1 457	3

注：μ：评估满意度；w：总等待时间；d：总距离；u：总车辆数。

基于上述基本设置，种群从10到50间变化，每种情况下10次运行平均结果如表10.4所示。结果表明，种群大于30时，种群数对遗传算法的性能没有显著影响。

表10.4 不同种群大小的结果

pop_size	μ	w	d	u
10	0.909	40.04	977	10
20	0.908	17.42	811	5
30	0.893	14.50	1 619	3
40	0.876	14.57	1 537	3
50	0.889	14.48	1 586	3

基于上述设置，目标函数(10.29)的权重变化见表10.5，它们对最终决策的影响见表10.6(10次运行的结果)。

表 10.5 不同权重

情况	ρ_1	ρ_2	ρ_3	ρ_4
1	0.85	0.05	0.05	0.05
2	0.05	0.85	0.05	0.05
3	0.05	0.05	0.85	0.05
4	0.05	0.05	0.05	0.85
5	0.50	0.30	0.10	0.10

表 10.6 不同权重的结果

情况	μ	w	d	u
1	0.854	16.20	1 442	3
2	1.000	53.51	921	17
3	0.843	14.89	1 557	3
4	0.976	54.41	822	19
5	0.881	15.75	1 467	3

从结果可知，任何对满意度(情况 2，$\rho_2=0.85$)或总行进距离(情况 4，$\rho_4=0.85$)目标的强调将导致车队大小和总等待时间的增加。

这里进一步将所提出的遗传算法与修改的插入启发式进行对比(因为 Solomon 的插入启发式不能处理模糊数据，所以用推-碰撞-挪过程加强)。表 10.7 中的结果表明遗传算法明显优于修改的插入启发式。

表 10.7 遗传算法与启发式的对比

问题	方法	μ	w	d	u
1	GA	0.779	11.83	1 436	4
	H	0.694	5.7	1 830	5
2	GA	0.866	12.05	1 367	4
	H	0.660	13.20	1 756	5
3	GA	0.799	15.26	1 405	3
	H	0.610	12.00	1 685	4
4	GA	0.765	5.78	1 631	3
	H	0.460	6.30	1 632	5
5	GA	0.728	11.32	1 322	3
	H	0.480	8.10	1 988	4
6	GA	0.871	15.80	1 428	3
	H	0.700	10.50	1 699	3

注：GA：遗传算法；H：修改的插入启发式。

10.3 布局-分配问题

布局-分配(location-allocation)问题也称作多韦伯(multi-Weber)问题或 p 中位(p-median)问题,这类问题来源于许多实际情况[111]。在欧几里得(Euclidean)空间上典型的单韦伯(single Weber)问题是寻找一个位置,使从代表顾客位置的一些固定点到它的距离和最小。Cooper 是正式认识并描述多韦伯问题的第一位学者。他证明目标函数既不是凹的也不是凸的,并存在许多局域最优[85]。Eilon 等的研究表明对于 $m=5,n=50$ 的单一问题,进行 200 次实验,可找到 61 个局域最小,并且最差解偏离最好解40.9%[117]。考虑到问题的组合规模(类型 II 的 Sterling 数),Cooper 提出了一种称作选择布局-分配(alternative location-allocation)的启发式,这是一种最好的启发方法。在过去的 20 年中,随着非线性规划技术的发展,通过松弛整数分配约束,同时考虑布局变量和分配变量,产生了一些新的方法。Murtagh 和 Niwattisyawong 提出了一种松弛 0-1 分配约束的方法。这种方法允许在[0,1]区间上取值,然后用大规模非线性规划软件包 MINOS 来解决布局和分配[304]问题。Bongartz 等提出的方法,也松弛 0-1 分配约束,同时还采用活动集合(active set)方法,并利用了布局-分配问题固有的特殊结构[45]。这些方法可用于具有几百个顾客和数十个设施的相当大规模的实际问题。但是,它们都不能保证全局最优和近全局最优。

Harris 等发现当可能解的数量是一个类型 II 的 Sterling 数时,可行解的数量相当少[209]。他们发现任何可行解中顾客的子集必须包括在非重叠凸域内,他们提出了一种算法,该算法利用问题这一凸域特性产生所有可行解,并通过对这些解的完全枚举找到最优解。Ostresh 提出了基于通行线(passing line)解决两设施韦伯问题的不同算法[323]。最近,Rosing 基于 Harris 的思想提出了关于一般多韦伯问题的优化方法[359]。然而,这些优化方法只能用于小规模问题。

Gong,Gen,Xu 和 Yamazaki 提出了一种解决具有能力约束的布局-分配问题的混和进化方法[177,465,466],这种方法将一些有效的传统优化技术用于分配子问题,并用这些技术处理能力约束。这一节中,我们将介绍他们的工作。

10.3.1 布局-分配模型

有 m 个设施需要布局,n 个已知位置的顾客分配给不同的设施,每个顾客的需求为 $q_j, j=1,2,\cdots,n$;每个设施具有的能力为 b_i, $i=1,2,\cdots,m$。我们需要找到设施布局和顾客对设施的分配,使顾客和服务他们的设施间的距离总和最小。见图 10.16。

这一问题的数学描述如下:

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij}\cdot\sqrt{(x_i-\bar{x}_j)^2+(y_i-\bar{y}_j)^2} \tag{10.31}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} z_{ij}=1; \quad j=1,2,\cdots,n \tag{10.32}$$

$$\sum_{j=1}^{n} q_j\cdot z_{ij}\leqslant b_i; \quad i=1,2,\cdots,m \tag{10.33}$$

$$z_{ij} = 0 \text{ 或 } 1; \quad i = 1,2,\cdots,m; \quad j = 1,2,\cdots,n \tag{10.34}$$

其中：

$(\bar{x}_j, \bar{y}_j)$——顾客 j 的位置，$j=1,2,\cdots,n$；

(x_i, y_i)——设施 i 的位置，决策变量，$i=1,2,\cdots,m$；

z_{ij}——0-1 决策变量：

$$z_{ij} = \begin{cases} 1, & \text{顾客 } j \text{ 由设施 } i \text{ 服务} \\ 0, & \text{其他} \end{cases}$$

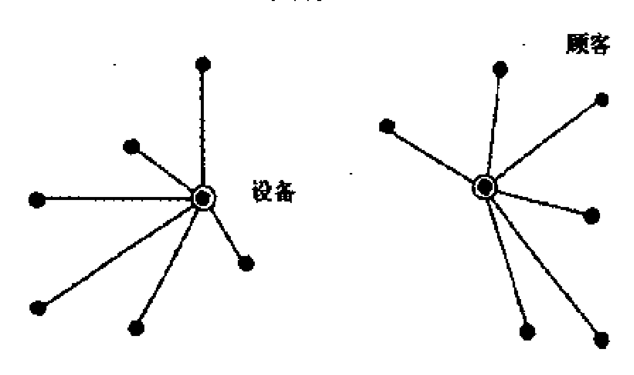


图 10.16 布局-分配问题

约束(10.32)保证每个顾客只由一个设施服务；约束(10.33)保证不超过每个设施的服务能力。

上述问题定义是一般布局-分配问题的扩充，其中，设施受服务能力的约束。分配子问题是一个一般的指派问题，它是一个已知 NP 难问题，不能简单地像对不受能力约束的问题一样用最近服务规则求解。对于这个问题，一般的选择布局-分配方法见文献[177]。选择布局-分配(alternative location-allocation——ALA)方法对于初始设施布局很敏感，由于它通过固定顾客分配来改进布局可能有时找不到全局最优或近全局最优。

10.3.2 混和进化方法

采用混和进化方法，问题分为两层：布局和分配。在分配层，因为这一子问题需经常解决，所以采用拉格朗日松弛法解决这个一般指派问题，以尽快获得解。在布局层，采用进化方法搜索整个布局区域。通过这种方法，整个可布局区域可以被有效地搜索以找到全局或近全局解。

1. 染色体表达方式

因为布局变量是连续的，所以采用浮点值染色体。染色体表达如下：

$$\mathbf{c}^k = [x_1^k, y_1^k, x_2^k, y_2^k, \cdots, x_m^k, y_m^k]$$

其中(x_i^k, y_i^k)是第 i 个设施在第 k 个染色体中的位置，$i=1,2,\cdots,m$。

2. 初始化

由于最优布局应在包括所有顾客的矩形区域内，因此设施初始布局是从这个矩形区域内任取位置，重复这一过程，直到产生所有种群成员。

初始化过程

步骤 1：寻找矩形$[x_{min}, x_{max}]\times[y_{min}, y_{max}]$；

步骤 2：令 $k=0$；

步骤 3：如果 $k > pop_size$ 停止，否则继续；

步骤 4：令 $k=k+1$，$i=0$；

步骤 5：如果 $i > m$ 则转步骤 3，否则继续；

步骤 6：在区间$[x_{min}, x_{max}]$上随机取 x_i^k；

步骤 7：在区间$[y_{min}, y_{max}]$上随机取值 y_i^k；

步骤 8：令 $i=i+1$，转步骤 5。

3. 评估

将顾客和为他服务的设施间的优化分配距离之和 $D(\mathbf{c}^k)$作为染色体 $\mathbf{c}^k$ 的适值函数。$D(\mathbf{c}^k)$的计算需要解决分配子问题。可用拉格朗日松弛法解决分配子问题，见文献[177]。

4. 交叉

交叉依赖于如何选择双亲及双亲如何繁殖后代。通常采用两种交叉策略：一个是自由交叉(free mating)，即任选两个双亲；另一个是优势交叉(dominating mating)，用最适合的个体作为一个固定的双亲，从种群中任选一个个体作为另一个双亲。这两种策略在进化过程中交替使用。

假设选择具有如下染色体的两个双亲产生后代：

$$\mathbf{c}^{k_1} = [x_1^{k_1}, y_1^{k_1}, x_2^{k_1}, y_2^{k_1}, \cdots, x_m^{k_1}, y_m^{k_1}]$$
$$\mathbf{c}^{k_2} = [x_1^{k_2}, y_1^{k_2}, x_2^{k_2}, y_2^{k_2}, \cdots, x_m^{k_2}, y_m^{k_2}]$$

只能产生一个后代：

$$\mathbf{c} = [x_1, y_1, x_2, y_2, \cdots, x_m, y_m]$$

后代染色体中的基因由下面等式确定：

$$x_i = r_i \cdot x_i^{k_1} + (1.0 - r_i) \cdot x_i^{k_2}$$
$$y_i = r_i \cdot y_i^{k_1} + (1.0 - r_i) \cdot y_i^{k_2}$$

其中，$r_1, r_2, \cdots, r_m$ 是(0,1)内的独立随机数。

交叉过程

步骤 1：令 $l=0$；

步骤 2：如果 $l \geqslant child_size \cdot (1-p_m)$则停止，否则 $l=l+1$ 继续；

步骤 3：如果第 t 代是奇数，转步骤 4，否则，转步骤 5；

步骤 4：从种群中任选两个双亲产生后代，然后转步骤 2；

步骤 5：任选一个双亲，另一个双亲则选最适合的个体，产生一个后代，然后转步骤 2。

5. 变异

这里提出两类变异算子，一类对双亲染色体进行较小的随机改变，产生一个新的后代染色体，称作精细变异(subtle mutation)；另一类采用与初始化过程相同的方法产生后代，称作强烈变异(violent mutation)。这两类变异算子在进化过程中交替进行。

假设变异候选染色体如下：

$$\mathbf{c}^k = [x_1^k, y_1^k, x_2^k, y_2^k, \cdots, x_m^k, y_m^k]$$

则由精细变异产生的后代 $\mathbf{c}=[x_1, y_1, x_2, y_2, \cdots, x_m, y_m]$确定如下：

$$x_i = x_i + [-\epsilon, \epsilon] \text{ 上的随机值}$$

$$y_i = y_i + [-\epsilon, \epsilon] \text{ 上的随机值}$$

其中 ϵ 是一个小正实数。由强烈变异产生的后代 $\mathbf{c}=[x_1, y_1, x_2, y_2, \cdots, x_m, y_m]$确定如下：

$$x_i = [x_{\min}, x_{\max}] \text{ 上的随机值}$$

$$y_i = [y_{\min}, y_{\max}] \text{ 上的随机值}$$

变异过程

步骤 1：设 $l=0$；

步骤 2：如果 $l \geqslant child_size \cdot p_m$则停止，否则 $l=l+1$ 继续；

步骤 3：任选一个双亲；

步骤 4：如果第 t 代是奇数，转步骤 5，否则，转步骤 6；

步骤 5：用精细变异产生后代，然后转步骤 2；

步骤 6：用强烈变异产生后代，然后转步骤 2。

6. 选择

用 ES-($\mu+\lambda$)选择方法在双亲和他们的后代中选出较好的个体形成下一代[10,14]。然而，这一策略通常导致进化过程退化。为了避免退化，提出了一种新的称作相对禁止(relative prohibition)的选择策略。

给定两个正参数 α 和 γ，染色体 $\mathbf{s}_k$ 的邻域定义如下：

$$\Omega(\mathbf{s}_k, \alpha, \gamma) \triangleq \{\mathbf{s} \mid \|\mathbf{s} - \mathbf{s}_k\| \leqslant \gamma, D(\mathbf{s}_k) - D(\mathbf{s}) < \alpha, \mathbf{s} \in R^{2m}\} \quad (10.35)$$

在选择过程中，一旦 $\mathbf{s}_k$ 被选入下一代，就禁止对它的邻域染色体进行选择。γ 值定义了 $\mathbf{s}_k$ 布局意义上的邻域，用于避免选择在布局上只有很小不同的个体。α 值定义了 $\mathbf{s}_k$ 适值意义上的邻域，用于避免选择在适值上只有很小差别的个体。等式(10.35)定义的邻域称作禁止邻域(prohibited neighborhood)。

选择过程

步骤 1：令所有双亲和后代是活动的。在双亲和后代中选择最好个体作为下一代第一个成员，选定的个体设为不活动的，并令 $l=1$。

步骤 2：如果 $l > pop_size$ 则停止；否则继续。

步骤 3：如果双亲或后代中没有活动的，在下一代中随机产生(pop_size-l)个新成员并停止；否则，继续。

步骤 4：在剩余活动的双亲和后代中选择最好个体。

步骤 5：检查这些个体是否落在选出成员的禁止邻域中，如果没有，将它作为下一代的新成员，并令 $l=l+1$。

步骤 6：将选出的个体改为不活动的，转步骤 3。

10.3.3 数值实例

用 Cooper 和 Rosing 的例子来测试混和进化方法的有效性[359]。Cooper 细心构造了包括三个自然组的前一半数据，Rosing 在任意点增加顾客数。这些例子为测试提出的

方法的有效性提供了一个好的标准，因为它们的全局优化解已知。

这些实例包括 30 个顾客，他们的位置坐标见表 10.8。顾客需求看作是相同的，并假设设施没有能力约束。

表 10.8 Cooper 和 Rosing 的实例的坐标

序号	X	Y	序号	X	Y
1	5	9	16	53	8
2	5	24	17	1	34
3	5	48	18	33	8
4	13	4	19	3	26
5	12	19	20	17	9
6	13	39	21	53	20
7	28	37	22	24	17
8	21	45	23	40	22
9	25	50	24	22	41
10	31	9	25	7	13
11	39	2	26	5	17
12	39	16	27	39	3
13	45	22	28	50	50
14	41	30	29	16	40
15	49	31	30	22	45

用 ALA 方法和混和进化方法(HEM)来解决这一问题。当用 ALA 方法时，由随机产生初始布局开始，对于每个例子，运行 40 次。HEM 的环境参数为：最大代数 $max_gen=600$，种群大小 $pop_size=40$，后代种群大小 $child_size=80$，变异率 $p_m=10\%$，$\alpha(t)=1.0$，$\gamma=1.0$。计算结果见表 10.9。在表中，误差百分比计算如下：(实际值-优化值)/优化值×100%。Rosing 没有给出 $n=30,m=6$ 情况下的结果，因为他估计了在 Convex 210 上 UNIX操作系统下它的计算时间将超过 10 CPU 小时。这种情况下 HEM 的计算时间在 4 分钟左右。如表 10.9 所示，HEM 通常可以找到全局最优，并在所有例子上超过 ALA。

表 10.9 Cooper 和 Rosing 实例的结果对比

问题 n/m	Rosing 方法的最优目标	ALA		HEM	
		最好	%误差	最好	%误差
15/2	214.281	219.2595	2.32	214.2843	0.0015
15/3	143.197	144.8724	1.17	143.2058	0.0061
15/4	113.568	115.4588	1.69	113.5887	0.0182
15/5	97.289	99.4237	2.19	97.5656	0.2843
15/6	81.264	84.0772	3.46	83.0065	2.14
30/2	447.728	450.3931	0.5952	447.73	0.0004
30/3	307.372	310.3160	0.9578	307.3743	0.0007
30/4	254.148	258.4713	1.7010	254.2246	0.0301
30/5	220.057	226.8971	3.1083	220.4335	0.1711
30/6	—	208.4301	3.4940	201.4031	0.0

10.4 障碍布局-分配问题

前一节给出的布局-分配模型是在没有考虑障碍的情况下对设施布局进行设计的理想情况。实际问题中，通常需要考虑障碍或禁止区域约束。例如湖泊、公园、建筑、河流等对设施布局设计形成的障碍。障碍有两类：

(1) 禁止布局；

(2) 禁止连接路径。

前一种情况，新设施不能布置在障碍区域内，这一问题也称作可憎的布局(obnoxious location)。后一种情况，顾客和设施的连接路径不能通过障碍区域，这一问题也称作移动障碍布局。当考虑障碍约束时，布局-分配问题求解变得更为复杂和困难。由于障碍布局-分配问题应用广泛，许多学者都集中于这一问题的研究。研究重点是减小最优解所在的可行布局集合的大小，Hakimi 指出当整个问题限制在一个给定网络内，且顾客只布置在节点上时，布局-分配问题通常具有设施布置在节点上的最优解[199]。这一结论将问题从连续搜索减化为组合问题。这一方法依赖于布局可能候选者的选择，这一候选者很难给出。

人们已充分地讨论了曼哈坦计量(Manhattan metric)l_1(或直线计量)障碍布局-分配问题，这是因为：①最优解可限制在候选点的特定有限集合内，其中候选点通过检查确定；②它有广泛的应用领域，如城市设施建筑、通信网络设计、设备布局和电路板设计，甚至机器人布局和路径[138,260,278]。Larson 和 Sadig 讨论了在存在行进障碍时具有 l_1 计量的设备布局问题[261]。Hamacher 和 Nickel 给出了设备布局问题一些在理论方面的结论和算法，其中只有设备布局受限[204]。

实际中常使用的距离计量是欧几里得(Euclidean)距离 l_2。然而，采用这一距离计量，障碍布局问题很难解决。布局变量的决策空间是连续的，目标函数变得非常复杂，难以计算斜率信息。Katz 和 Cooper 讨论了只有一个禁止范围的 l_2 距离设备布局问题[242]。文献[178]中讨论了具有多边形障碍的布局-分配问题。Gong，Gen，Xu 和 Yamazaki 将他们的工作扩展到障碍布局-分配问题[175,176]。因为所用的进化方法的许多过程与前一节的讨论相同，所以这一节中主要讨论不同部分。

10.4.1 障碍布局-分配模型

有 n 个位置已知的顾客和 m 个设施，设施对所有顾客提供某种服务，例如，提供物料或能源，此外还存在 p 个代表禁止区域的障碍。为进行数学描述，做如下假设：

(1) 顾客 j 的服务需求为 q_j，$j=1,2,\cdots,n$；

(2) 设备 i 的服务能力为 b_i，$i=1,2,\cdots,m$；

(3) 每个顾客只由一个设施服务；

(4) 新设施不能建在任何障碍区内；

(5) 设施和顾客间的连接路径不允许通过任何障碍。

问题是要选择设施的最好布局，使顾客与他们的服务设施间的距离总和为最小，如图

10.17所示。

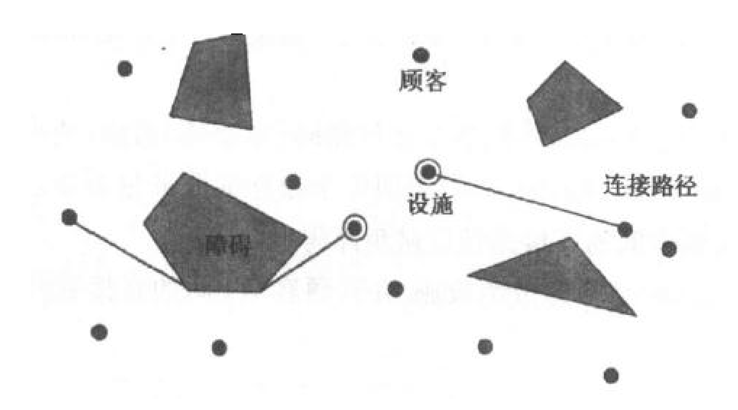


图 10.17　障碍布局-分配问题

假设所有的障碍可描述为凸多边形，障碍布局-分配问题可用下述三级模型描述：

第一级

$$\min \quad D(S_1,S_2,\cdots,S_m) \tag{10.36}$$

$$\text{s.t.} \quad S_i \bar{\in} inner(P_l); \quad i=1,2,\cdots,m; \quad l=1,2,\cdots,p \tag{10.37}$$

第二级

$$D(S_1,S_2,\cdots,S_m) \triangleq$$

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} d(S_i,W_j)\cdot z_{ij} \tag{10.38}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} z_{ij}=1; \quad j=1,2\cdots n \tag{10.39}$$

$$\sum_{j=1}^{n} q_j\cdot z_{ij}\leqslant b_i; \quad i=1,2\cdots m \tag{10.40}$$

$$z_{ij}=0 \text{ 或 } 1; \quad i=1,2,\cdots,m; \quad j=1,2,\cdots,n \tag{10.41}$$

第三级

$$d(S_i,W_j) \triangleq \min \mid l \mid \tag{10.42}$$

$$\text{s.t.} \quad l \in L(S_i,W_j);\ i=1,2,\cdots,m;\ j=1,2,\cdots,n \tag{10.43}$$

其中：

$W_j=(\bar{x}_j,\bar{y}_j)$——第 j 个顾客的位置；

$(\bar{x}_j,\bar{y}_j)$——顾客位置的 x 坐标和 y 坐标；

P_l——第 l 个障碍(多边形)；

$inner(P_l)$——在第 l 个障碍内的点集合，$l=1,2,\cdots,p$；

$S_i=(x_i,y_i)$——第 i 个设施的位置；

(x_i,y_i)——设施位置的 x 坐标和 y 坐标，决策变量；

$D(S_1,S_2,\cdots,S_m)$——设施在某个固定位置时，所有顾客到服务于他的设施的最小距离和；

$d(S_i,W_j)$——在避免障碍情况下，设施 S_i 和顾客 W_j 间最短连接路径；

z_{ij}——0-1 决策变量，$z_{ij}=1$ 表明第 j 个顾客由第 i 个设施服务，否则 $z_{ij}=0$；

$L(S_i,W_j)$——在避免任何障碍 P_l 的情况下，连接第 i 个设施和第 j 个顾客的可能路径集合。

约束(10.37)表明设备布局不能落在任何障碍(多面体)内部；约束(10.39)表明每个顾客只能由一个设施服务；约束(10.40)表明每个设施的服务量不能超过它的能力；约束(10.43)表明设备和顾客间的连接路径应避免障碍。

如果忽略障碍，$d(S_i,W_j)$即成为设施 S_i 和顾客 W_j 间的直接欧几里得距离，其表示如下：

$$d(S_i,W_j)=\sqrt{(x_i-\bar{x}_j)^2+(y_i-\bar{y}_j)^2} \tag{10.44}$$

当考虑障碍时，$d(S_i,W_j)$是 S_i，W_j 和 P_l 的函数，$l=1,2,\cdots,p$。这一函数不能用数学描述直接表达。

这个模型是三级混和整数规划问题，目标函数和约束非常复杂，有些甚至不能以数学方式直接描述。目标函数梯度信息也难以获得，而且有许多局域最优解。传统方法不能很好地解决这一问题，进化方法由于能够进行全局最优搜索和需要较少的计算条件，因而更适合于用来寻找全局或近全局最优。

10.4.2 可行布局

第一级约束要求所有设施布局必须在障碍区域之外，对这一约束的检查等价于检查计划点是否落在多面体内。这可以通过计算所有由障碍多面体一边和计划点构成的所有三角形的面积与障碍多面体的面积进行对比来完成。算法见文献[177]。

10.4.3 避免障碍的最短路径

在第三级中，需要找到绕开所有障碍物连接两个特定点的最短路径。许多学者进行了寻找避免障碍物的路径的研究工作，特别是在机器人导航方面。这里采用了可视图的思想，然而，加人了一些不同的想法以开发出适应这类问题的特定算法。根据经验，只有少数障碍物与计划中连接特定两点的连接路径有关，因而不必建立包括所有障碍的很大规模的整体可视图。只考虑相关障碍建立小规模可视图的算法见文献[177]，然后用 Dijikstra 最短路径算法解可视图，即可得到避免障碍的最短路径。

10.4.4 混和进化方法

解决障碍布局-分配问题的进化方法与解决能力约束布局-分配问题的进化方法除两点不同外，其他基本相同。其中一点是染色体的可行性调整，另一点是计算能够避开障碍物的、连接顾客和其服务设施的最短路径。

1. 染色体可行性调整

由于存在障碍，由初始化、交叉和变异产生的染色体布局可能是不可行的。通常，有三种方法处理不可行染色体。第一种方法是丢掉它，但根据其他学者的经验，这种方法会降

低效率;第二种方法是对不可行染色体加以惩罚;第三种方法是根据特定问题的特点修复不可行染色体。这里,基于障碍布局-分配问题的特点,可用一种简单的方法修复不可行染色体。首先,检查布局的可行性,一旦一个布局不可行,即用它所在障碍物中离它最近的那个点替换它,使它变成可行的。如图 10.18 所示。

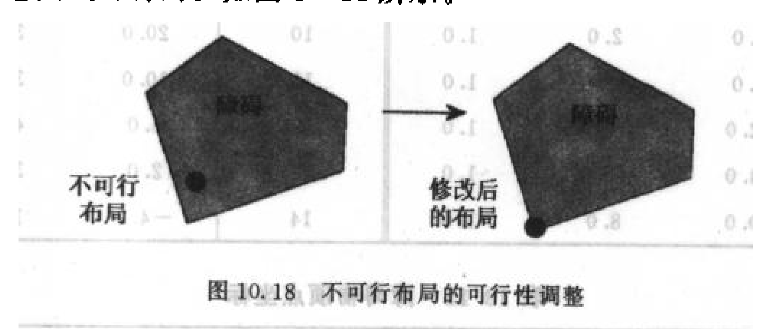


图 10.18 不可行布局的可行性调整

2. 算法

混和进化方法的算法描述如下:

混和进化方法过程

begin

  初始化;

  调整染色体可行性;

  评估染色体;

  for $i=1$ **to** 最大代数 *max_gen* **do**

    begin

    交叉;

    变异;

    调整染色体可行性;

    评估;

    选择;

  end

end

10.4.5 实例研究

实例问题为处理油田电站的布局问题。有 14 座油井,它们的位置坐标和对电的需求见表 10.10。有四个障碍物:两个小镇、一个大湖泊和一个大的工厂(见表 10.11)。需要寻找三个电站的位置,其中两个电站可以分别为四座油井提供电力,一个电站可以为六座油井提供电力(见表 10.12)。电站不能建在障碍物上,出于安全和工程可行性,电线不能通过障碍物。

表 10.10 油田坐标和需求

j	x_j	y_j	q_j	j	x_j	y_j	q_j
1	−2.0	0.0	1.0	8	22.0	20.0	1.0
2	2.0	4.0	1.0	9	26.0	30.0	1.0
3	8.0	2.0	1.0	10	20.0	38.0	1.0
4	5.0	12.0	1.0	11	10.0	36.0	1.0
5	12.0	14.0	1.0	12	6.0	40.0	1.0
6	24.0	16.0	1.0	13	2.0	36.0	1.0
7	30.0	8.0	1.0	14	−4.0	12.0	1.0

表 10.11 障碍物顶点坐标

障碍	障碍 1	障碍 2	障碍 3	障碍 4
顶点数	5	3	4	5
障碍顶点坐标	(2.0,1.0)	(8.0,6.0)	(0.0,8.0)	(7.0,15.0)
	(6.0,1.0)	(10.0,3.0)	(3.0,7.0)	(16.0,18.0)
	(8.0,4.0)	(9.0,12.0)	(5.0,14.0)	(14.0,32.0)
	(4.0,6.0)		(0.5,16.0)	(12.0,34.0)
	(1.0,3.0)			(5.0,23.0)

表 10.12 电站供电能力

电站 i	能力 b_i
1	4.0
2	6.0
3	4.0

对这一问题采用了六个连续决策变量描述三个电站坐标位置，并用 42 个 0-1 决策变量描述油井对电站的分配。采用上述混和进化方法获得的解比文献[177]中用启发式算法获得的解更满意。

混和方法的环境参数为：$max_gen=1200$，$pop_size=10$，$child_size=40$，$p_m=1.0$，$\alpha=1.0$，$\gamma=4.0$。结果对比见表 10.13。进化过程见图 10.19。

表 10.13 HEM 和启发式方法的结果对比

方法	距离和	设施布局	顾客分配
HEM	97.435	1. (8.9509,36.9761)	10, 11, 12, 13
		2. (3.1611,6.3528)	1, 2, 3, 4, 5, 14
		3. (23.9536,17.3541)	6, 7, 8, 9
启发式	107.0858	1. (9.5,37.5)	10, 11, 12, 13
		2. (1.0,4.59)	1, 2, 3, 14
		3. (19.8333,16.6667)	4, 5, 6, 7, 8, 9



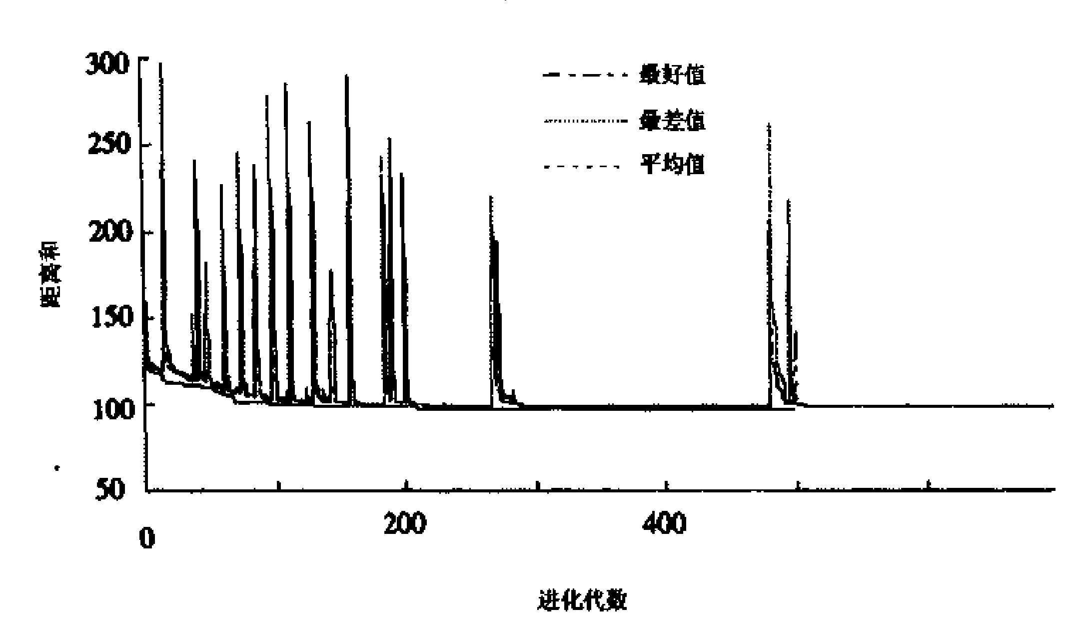


图 10.19 HEM 的进化过程

可以看出，在这一实例中，在寻找更好解方面，所提出的混和进化方法比启发式方法更为有效。

10.5 生产计划问题

生产计划问题是在不同需求和费用情况下，确定生产速率。许多学者对这一问题进行了研究，如 Federgruen 和 Schechner[125]，Liu[277]，Luss[280]，Moinzadeh 和 Nahmias[297] 以及 Wang，Wilson 和 Odrey[411]。以前的工作主要是解决离散时间生产计划问题，解决连续时间情况的方法较少。最近 Gen 和 Liu 研究了连续时间的生产计划问题，并提出了一种解决这一问题的进化方法[161，444]。

10.5.1 生产计划问题描述

为准确描述这一问题，先介绍一些概念：

$z(t)$——在时间 t 的生产速率；

$r(t)$——在时间 t 的需求率；

$y(t)$——在时间 t 的库存水平；

$c(z)$——生产速率为 z 时单位时间生产费用；

$g(dz/dt)$——生产速率改变的单位时间费用；

$l(y)$——库存水平为 y 时单位时间库存费用。

如果 $y<0$，$l(y)$是未交付定货量为 y 时的费用；如果 $y>0$，$l(y)$是存储量为 y 时的单位时间的存储费用。

时间 t 的生产总量为

$$Z(t) = \int_0^t z(\tau)d\tau \tag{10.45}$$

时间 t 的总需求为

$$R(t) = \int_0^t r(\tau)d\tau \tag{10.46}$$

则有

$$y(t) = Z(t) - R(t) \tag{10.47}$$

连续时间生产计划问题是最小化

$$J(z) = \int_0^T \left\{ c(z(t)) + g\left(\frac{dz(t)}{dt}\right) + l(y(t)) \right\} dt \tag{10.48}$$

即制定生产计划 $z(t)$（时间 t 的函数），使 $J(z)$ 为最小。

在这个模型中，我们假设生产速率在任何连续区间上可以是不同的。在解释费用函数 $J(z)$ 的积分 $\int_0^T g(dz/dt)dt$ 时，需要注意的是我们不希望限定生产策略函数是可微的甚至是连续的。如果 $z(t)$ 在某些点上是不连续的，则 dz/dt 不能定义，且差分 $g(z(t+0))-g(z(t-0))$ 将被看作是那一点的积分基值。因此，该积分是连续的 $z(t)$ 的积分与不连续点的积分基值的和。

如果每次改变生产速率时需要一个正的装设费用，则优化生产速率必须是一个阶梯函数(step function)，即生产速率在有限时间内改变，且在每个时间区间内是常数。

10.5.2 生产计划问题评价

1. *表达方式*

如上所述，优化生产计划是一个阶梯函数的形式，因而可重写如下：

$$z(t) = \begin{cases} z_i, & t_{i-1} \leqslant t < t_i; \quad i = 1,2,\cdots,n \\ 0, & t > T \end{cases} \tag{10.49}$$

让我们看看图 10.20 给出的例子。

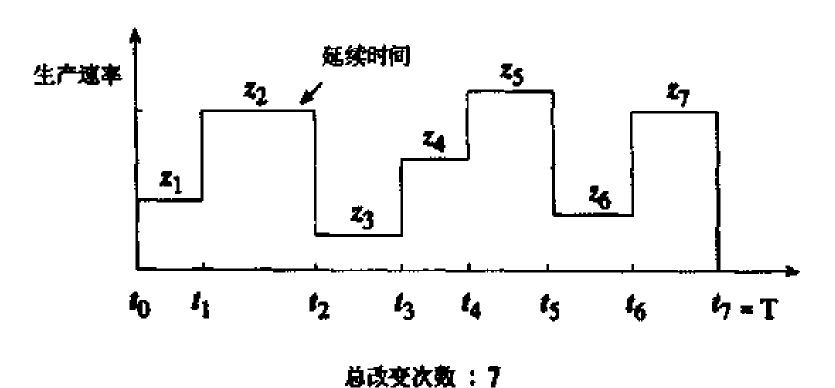


图 10.20 计划期上生产速率的例子

从图中我们可以看出，生产计划由下述三个因素描述：

(1) 计划展望期 T 上生产速率改变的总次数 n，它将计划展望期分成几个时段；

(2) 在 T 上每个时段的生产速率值 z_i；

(3) 每个时段的延续时间 $t_i - t_{i-1}$，在这段时间内生产速率保持不变。

一个染色体必须具有生产计划的三个因素以便表达问题的解。问题解决之前，总改变次数 n 是未知的，为适应这种情况，用一个变长度串 $[x_1, x_2, \cdots, x_{2n}]$ 作为表达问题解的染色体，其中，偶数下标元素 x_{2i}，$i=1,2,\cdots,n$ 代表生产速率，奇数下标元素 x_{2i-1}，$i=1,2,\cdots,n$ 代表时段 i 延续时间与计划展望期 T 的比率。

从染色体得到的解如下：

$$
\begin{aligned}
t_0 &= 0 \\
t_i &= \frac{x_1 + x_3 + \cdots + x_{2i-1}}{x_1 + x_3 + \cdots + x_{2n-1}} \cdot T; \qquad i = 1,2,\cdots,n \\
z_i &= x_{2i}; \quad i = 1,2,\cdots,n
\end{aligned}
\tag{10.50}
$$

从等式(10.50)容易看出染色体总能产生可行解。

2. 初始化

初始种群由下述步骤随机产生：

步骤 1：随机产生整数 n 作为生产速率的改变次数，通常，n 的可能范围可以由特定问题的知识给出。

步骤 2：在区间 $[\varepsilon,1]$ 上产生 n 个随机数 $x_1, x_3, \cdots, x_{2n-1}$。

步骤 3：在区间 $[\varepsilon, up_bound]$ 上产生 n 个随机数 $x_2, x_4, \cdots, x_{2n}$。

步骤 4：重复上述步骤，产生 *pop_size* 个染色体。

其中 ε 是小的正数，*up_bound* 是生产速率的可能上限。

3. 评估与选择

采用一类基于顺序的评估函数评定每个染色体的适值。评估过程主要包括三步：

步骤 1：根据目标(10.48)计算目标值。

步骤 2：将染色体根据目标值按升序排列。

步骤 3：对每个染色体赋给基于顺序的适值。令 r_k 为染色体 $\mathbf{v}_k$ 的位次。对于用户给定的参数 $a \in (0,1)$，基于顺序的适值函数定义如下：

$$eval(\mathbf{v}_k) = a(1-a)^{r_k - 1}$$

其中 $r_k = 1$ 代表最好染色体，$r_k = pop_size$ 代表最差染色体，于是有

$$\sum_{k=1}^{pop_size} eval(\mathbf{v}_k) \approx 1$$

选择过程基于转轮法，进行 *pop_size* 次，其中 *pop_size* 是种群数，每次有一个染色体被选进新的种群。

4. 交叉

采用定义为两个向量线性组合的算术交叉[283]。对于每对双亲 $\mathbf{v}_1$ 和 $\mathbf{v}_2$，如果他们具有相同的维数(相同的 n)，交叉算子产生两个后代 $\mathbf{v}'$ 和 $\mathbf{v}''$：

$$
\begin{aligned}
\mathbf{v}' &= c_1 \cdot \mathbf{v}_1 + c_2 \cdot \mathbf{v}_2 \\
\mathbf{v}'' &= c_2 \cdot \mathbf{v}_1 + c_1 \cdot \mathbf{v}_2
\end{aligned}
$$

其中 $c_1,c_2\geqslant 0$ 且 $c_1+c_2=1$。因为约束集合是凸的，如果双亲都是可行的，算术交叉操作可保证两个后代是可行的。

5. 变异

这里定义了两类变异算子，第一类是不同维数变异，产生一个新的维数 n，并且产生一个新的可行解$(x_1,x_2,\cdots,x_{2n})$替代原有的解。第二类是同维数变异，沿负梯度（或次梯度）方向产生染色体。

连续变化函数 f 的 Taylor 展开式为

$$f(\mathbf{x}+\Delta\mathbf{x})=f(\mathbf{x})+(\nabla f(\mathbf{x}+\theta\Delta\mathbf{x}))^T\Delta\mathbf{x},\qquad \mathbf{x}\in R^n$$

其中，$0\leqslant\theta\leqslant 1$；$\nabla f(\mathbf{x})$是函数 f 在点 $\mathbf{x}$ 的梯度，$\Delta\mathbf{x}$ 是 R^n 上小的波动。由于这一问题的复杂性，梯度可能不存在。这种情况下，我们可以由下述公式近似地计算梯度的第 i 个分量（不论是否存在）：

$$\frac{f(x_1,\cdots,x_i+\Delta x_i,\cdots,x_n)-f(x_1,\cdots,x_i,\cdots,x_n)}{\Delta x_i}\tag{10.51}$$

其中 Δx_i 是一个小的实数。

用 p_{m1}和 p_{m2}分别表示不同维数和同维数变异概率，变异过程进行如下：在[0,1]上随机产生实数 r，如果 $r<p_{m1}$，进行不同维数变异；如果 $p_{m1}<r<p_{m1}+p_{m2}$进行同维数变异。对同维数变异，近似负梯度方向 $\mathbf{d}$ 由公式(10.51)产生。设一个双亲为 $\mathbf{v}=(x_1,x_2,\cdots,x_{2n})$，产生一个后代如下：

$$\mathbf{v}'=\mathbf{v}+M\cdot\mathbf{d}$$

如果它不可行，令 M 是$(0,M)$间的任意实数，直到 $\mathbf{v}+M\cdot\mathbf{d}$ 可行为止。

10.5.3 例子

我们考虑下述例子：需求率是

$$r(t)=1+\sin\left(\frac{t}{T}4\pi\right),\qquad 0\leqslant t\leqslant T\tag{10.52}$$

单位时间生产费用函数为

$$c(z)=c\cdot z\tag{10.53}$$

单位时间生产速率改变费用函数为

$$g\left(\frac{dz}{dt}\right)=k+w\cdot\left|\frac{dz}{dt}\right|\tag{10.54}$$

单位时间存储费用为

$$l(y)=\begin{cases}h\cdot y, & y\geqslant 0\\ s\cdot(-y), & y<0\end{cases}\tag{10.55}$$

参数设置为：$T=100,c=5,w=20,k=100,h=3,s=10$，初始库存水平为 $y(0)=0$，$pop_size=50,p_c=0.1$，不同维数变异率 $p_{m1}=0.1$，同一维数变异率 $p_{m2}=0.4$，基于等级的评估函数中参数 $a=0.1$。

图 10.21 表明在 3 000 次迭代中最好目标值是 2370.78。最好生产计划具有五次生产改变，生产计划在区间[0,1.78)上生产速率为 1.93，在区间[1.78,27.57)上生产速率为

1.56，在区间[27.57，46.03)上生产速率为0.50，在区间[46.03，69.49)上生产速率为1.47，在区间[69.49，100]上生产速率为0.54，如图10.22所示。生产量和需求量的累加值见图10.23。

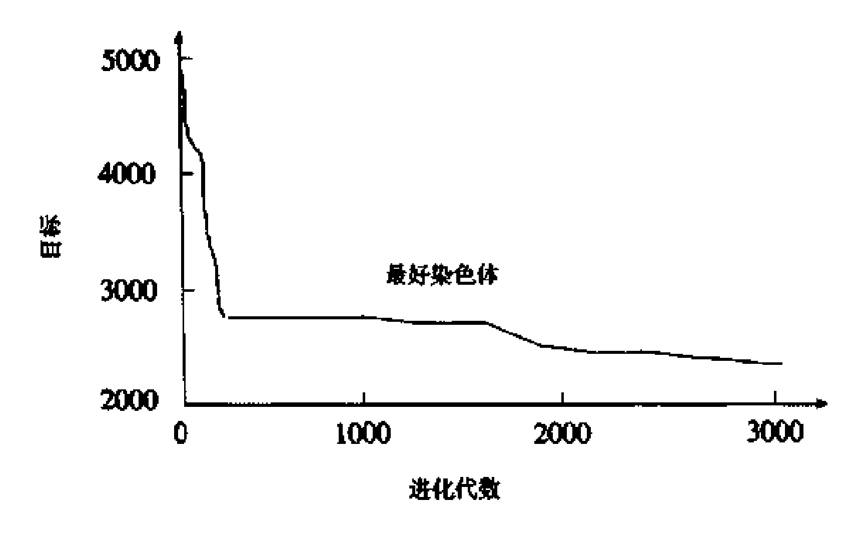


图10.21 进化过程

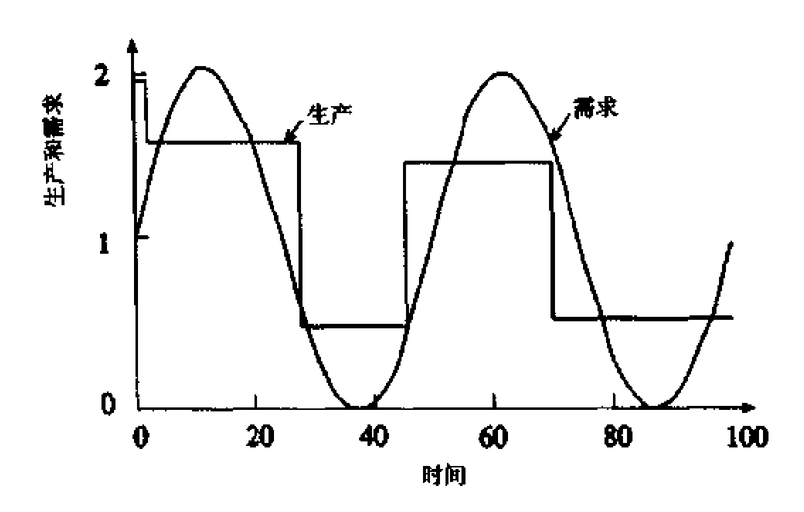


图10.22 生产率和需求率

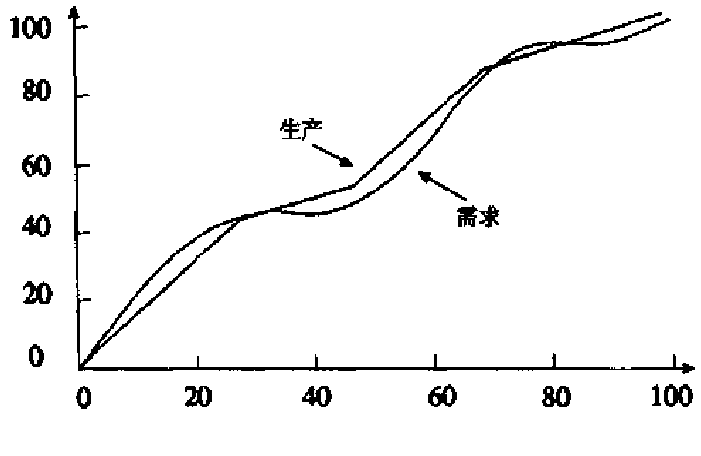


图10.23 生产量和需求量的累加值

参 考 文 献

[1] Abuali, F., R. Wainwright, and D. Schoenefeld, A new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem, in Eshelman [120], pp. 470—475.

[2] Ackley, D., *A Connectionist Machine for Genetic Hillclimbing*, Kluwer Academic Publishers, Boston, 1987.

[3] Adams, J., E. Balas, and D. Zawack, the shifting bottleneck procedure for job shop scheduling, *International Journal of Flexible Manufacturing Systems*, vol. 34, no. 3, pp. 391—401, 1987.

[4] Alander, J., Interval arithmetic and genetic algorithms in global optimization, In Pearson et al. [335], pp. 387—391.

[5] Albrecht, R., C. Reeves, and N. Steele, editors, *Artificial Neural Nets and Genetic Algorithms*, Springer-Verlag, New York, 1993.

[6] Alefeld, G. and J. Herzberger, *Introduction to Interval Computions*, Academic Press, New York, 1983 (translated by J. Rokne.)

[7] Alvarez-Valdés, R. and J. Tamarit, Heurisitic algorithms for resource constrained project scheduling: a review and an empirical analysis, in Slowinski, R. and J. Weglarz, editors, *Advances in Project Scheduling*, pp. 113—134, Elsevier Science Publishers, Amsterdam, 1989.

[8] Aneja, Y. and K. Nair, Bicriteria transportation problem, *Management Science*, vol. 25, pp. 73—78, 1978.

[9] Applegate, D. and W. Cook, A computational study of the job shop scheduling problem, *ORSA Journal of Computing*, vol. 3, no. 2, pp. 149—156, 1991.

[10] Bäck, T., Selective pressure in evolutionary algorithms: a characterization of selection mechanisms, in Fogel [132], pp. 57—62.

[11] Bäck, T., *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.

[12] Bäck, T., D. Fogel, and Z. Michalawecz, editors, *Handbook of Evolutionary Computation*, Oxford University Press, Oxford, 1997.

[13] Bäck, T. and F. Hoffmeister, Extended selection mechanisms in genetic algorithms, in Belew and Booker [30], pp. 92—99.

[14] Bäck, T. and S. Khuri, An evolutionary heuristic for the maximum independent set problem, in Fogel [132], pp. 531—535.

[15] Bagchi, S., S. Uckun, Y. Miyabe, and K. Kawamura, Exploring problem-specific recombination operators for job shop scheduling, in Belew and Booker [30], pp. 10—17.

[16] Bagchi, U., Simultaneous minimization of mean and variation of flow time and waiting time in single machine systems, *Operations Research*, vol. 37, pp. 118—125, 1989.

[17] Bagchi, U., L. Chang, and R. Sullivan, Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalties and a common due date, *Naval Research Logistics Quarterly*, vol. 34, pp. 739—751, 1987.

[18] Bagchi, U., R. Sullivan, and L. Chang, Minimizing mean absolute deviation of completion times about a common due date, *Naval Research Logistics Quarterly*, vol. 33, pp. 227—240, 1986.

[19] Baker, J., Reducing bias and inefficiency in the selection algorithm, in Grefenstette [186], pp. 14—21.

[20] Baker, J., Adaptive selection methods for genetic algorithms, in Grefenstette [189], pp. 100—111.

[21] Baker, K., *Introduction to Sequencing and Scheduling*, John Wiley & Sons, New York, 1974.

[22] Baker, K. and G. Scudder, Sequencing with earliness and tardiness penalties: A review, *Operations Research*, vol. 38, pp. 22—36, 1990.

[23] Balas, E., Machine sequencing via disjunctive graphs: an implicit enumeration algorithm, *Operations Research*, vol. 17, pp. 941—957, 1969.

[24] Barlow, R., L. Hunter, and F. Proschan, Optimum redundancy when components are subject to two kinds of failure, *Journal of SIAM*, vol. 11, pp. 64—73, 1963.

[25] Bauer, R., *Genetic Algorithms and Investment Strategies*, John Wiley & Sons, New York, 1994.

[26] Bazaraa, M., J. Jarvis, and H. Sherali, *Linear Programming and Network Flows*, 2nd ed., John Wiley & Sons, New York, 1990.

[27] Bazaraa, M., and O. Kirca, A branch and bound based heuristic for solving the QAP, *Naval Research Logistics*

Quarterly, vol. 30, pp. 287—304, 1983.

[28] Bazaraa, M., H. Sherali, and C. Shetty, *Nonlinear Programming: Theory and Algorithm*, 2nd ed., John Wiley & Sons, New York, 1993.

[29] Bean, J., Genetic algorithms and random keys for sequencing and optimization, *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154—160, 1994.

[30] Belew, R. and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[31] Bell, C. and J. Han, A new heuristic solution method in resource constrained project scheduling, *Naval Research Logistics*, vol. 38, pp. 315—331, 1991.

[32] Bertsimas, D., The probabilistic minimum spanning tree problem, *Networks*, vol. 20, pp. 245—275, 1990.

[33] Bhanu, B. and S. Lee, *Genetic Learning for Adaptive Image Segmentation*, Kluwer Academic Publishers, Norwell, MA, 1994.

[34] Biethahn, J., *Evolutionary Algorithms in Management Applications*, Springer-Verlag, Berlin, 1995.

[35] Bit, A., M. Biswal, and S. Allam, Fuzzy programming approach to multicriteria decision making transportation problem, *Fuzzy Sets and Systems*, vol. 50, pp. 135—141, 1992.

[36] Bits, A., M. Biswal, and S. Alam, Fuzzy programming approach to multiobjective solid transportation problem, *Fuzzy Sets and Systems*, vol. 57, pp. 183—194, 1993.

[37] Bjorndal, M., A. Caprara, P. Cowling, P. Croce, H. Lourence, F. Malucelli, A. Orman, D. Pisinger C. Rego, and J. Salazar, Some thoughts on combinatorial optimization, *European Journal of Operational Research*, vol. 83, pp. 253—270, 1995.

[38] Blackstone, J., D. Phillips, and G. Hogg, A state-of-the-art survey of dispatching rules for manufacturing job shop operations, *International Journal of Production Research*, vol. 20, pp. 26—45, 1982.

[39] Blanton, J. and R. Wainwright, Multiple vehicle routing with time and capacity constaints using genetic algorithm, in Forrest [137], pp. 452—459.

[40] Blazewicz, J., Complexity of computer scheduling algorithms under resource constraints, in *Proceedings, First Meeting of the AFCET-SMF on Applied Mathematics*, pp. 169—178, Palaiseau, Poland, 1978.

[41] Blazewicz, J., K. Ecker, G, Schmidt, and J. Weglarz, *Scheduling in Computer and Manufacturing Systems*, 2nd ed., Springer-Verlag, New York, 1994.

[42] Boctor, F., Some efficient multi-heuristic procedures for resource constrained project scheduling, *European Journal of Operational Research*, vol. 49, pp. 3—13, 1990.

[43] Bodin, L., B. Golden, A. Assad, and M. Ball, Routing and scheduling of vehicles and crews: the state of the art, *Computers and Operations Research*, vol. 10, pp. 62—212, 1983.

[44] Bolc, L. and J. Cytowski, *Search Methods for Artificial Intelligence*, Academic Press, London, 1992.

[45] Bongartz, I., A projection method for norm location-allocation problems, *Mathematical Programming*, vol. 66, pp. 283—312, 1994.

[46] Booker, L., Improving search in genetic algorithms, in Davis [102].

[47] Bracken, J., and G. McCormick, *Selected Applications of Programming*, John Wiley & Sons, New York, 1968

[48] Brindle, A., Genetic Algorithms for Function Optimization, Ph. D. thesis, University of Alberta, Edmonton, 1981.

[49] Bruno L., J. Coffman, and R. Sethi, Scheduling independent tasks to reduce mean finishing time, *Communications on ACM*, vol. 17, pp. 382—387, 1974.

[50] Bruns, R., Direct chromosome representation and advanced genetic operators for production scheduling, in Forrest [137], pp. 352—359.

[51] Budnick, F, D. McLeavey and R. Mojena, *Principles of Research for Management*, 2nd ed., Irwin Press, Homewood, IL, 1988.

[52] Burkard, R. and T. Bonninger, A heuristic for quadratic boolean program with applications to quadratic assignment problems, *European Journal of Operational Research*, vol. 13, pp. 374—386, 1983.

[53] Campbell, H., R. Dudek, and M. Smith, A heuristic algorithm for the n-job m-machine sequencing problem, *Management Science*, vol. 16B, pp. 630—637, 1970.

[54] Chambers, L., *Practical Handbook of Genetic Algorithms*, vols. 1 and 2, CRC Press, New York, 1995.

[55] Charnes, A. and W. W. Cooper, *Management Models and Industrial Applications of Linear Programming*,

John Wiley & Sons, New York, 1961.

[56] Chartrand, G., and L. Lesniak, *Graphs and Digraphs*, Wadsworth & Brooks, Monterey, CA, 1979.

[57] Chen, C., V. Vempati, and N. Aljaber, An application of genetic algorithms for flow shop problems, *European Journal of Operational Research*, vol. 80, pp. 389—396, 1995.

[58] Chen, S and C. Hwang, *Fuzzy Multiple Attribute Decision Making*, Springer-Verlag, Berlin, 1992.

[59] Cheng, C., Study on Optimal Design of Fuzzy Facility Layout, Master's thesis, Ashikaga Institute of Technology, Ashikaga, Japan, March 1995.

[60] Cheng, R. and M. Gen, On film copy deliverer problem, In Zheng, W., editor, *Proceedings of the Second International Conference on Systems Science and Systems Engineering*, pp. 542—547, Beijing, 1993.

[61] Cheng R. and M. Gen, Crossover on intensive search and traveling salesman problem, in Gen and Kobayashi [160], pp. 568—571.

[62] Cheng, R. and M. Gen, Evolution program for resource constrained project scheduling problem, in Fogel [132], pp. 736—741.

[63] Cheng, R. and M. Gen, Vehicle routing problem with fuzzy due-times using genetic algorithm, *in The Third Conference of Asian-Pacific Operational Research Society*, Fukuoka, Japan, 1994.

[64] Cheng, R. and M. Gen, Resource constrainted project scheduling problem using genetic algorithms, *International Journal of Intelligent Automation and Soft Computing*, 1996 (to appear).

[65] Cheng, R. and M. Gen, Fuzzy vehicle routing and scheduling problem using genetic algorithms, in Herrera, F. and J. Verdegay, editors, *Genetic Algorithms and Soft Computing*, pp. 683—709, Springer-Verlag, 1996.

[66] Cheng, R., M. Gen, and M. Sasaki, Film-copy deliverer problem using genetic algorithms, *International Journal of Computers and Industrial Engineering*, vol. 29, no. 1—4, pp. 549—553, 1995.

[67] Cheng, R. and M. Gen, Genetic algorithms for multi-row machine layout problem, *Engineering Design and Automation*, 1996 (to appear).

[68] Cheng, R. and M. Gen, Minmax earliness/tardiness scheduling in identical parallel machine syetem using genetic algorithm, *International Journal of Computers and Industrial Engineering*, vol. 29, no. 1—4, pp. 513—517, 1995.

[69] Cheng, R. and M. Gen, Vehicle routing problem with fuzzy due-time using genetic algorithms, *Japanese Journal of Fuzzy Theory and Systems*, vol. 7, no. 5, pp. 1050—1061, 1995.

[70] Cheng, R. and M. Gen, Genetic search for facility layout design under interflows uncertainty, *Japanese Journal of Fuzzy Theory and Systems*, vol. 8, no. 2, pp. 335—346, 1996.

[71] Cheng, R., M. Gen, and Y. Tsujimura, A tutorial survey of job-shop scheduling problems using genetic algorithms, part I. representation, *International Journal of Computers and Industrial Engineering*, vol. 30, no. 4, pp. 983—997, 1996.

[72] Cheng, T. and X. Cai, On the complexity of completion time variance minimization problem, Working paper, University of Western Australia, Perth, 1990.

[73] Cheng, T. and M. Gupta, Survey of scheduling research involving due date determination decision, *European Journal of Operational Research*, vol. 38, pp. 156—166, 1989.

[74] Cheng, T. and C. Sin, A state-of-the-art review of parallel-machine scheduling research, *European Journal of Operational Research*, vol. 47, pp. 271—292, 1990.

[75] Christofides, N., R. Alvarez-Valdés and J. Tamarit, Project scheduling with resource constrained: A branch and bound approach, *European Journal of Operational Research*, vol. 29, pp. 262—273, 1987.

[76] Cleveland, G. and S. Smith, Using genetic algorithms to schedule flow shop releases, in Schaffer [370], pp. 160—169.

[77] Climaco, J., C. Antunes, and M. Alves, Interactive decision support for multiobjective transportation problem, *European Journal of Operational Research*, vol. 65, pp. 58—67, 1993.

[78] Coffman, E., *Computer and Job-shop Scheduling Theory*, John Wiley & Sons, New York, 1976.

[79] Cohen, J., R. Church and D. Sheer, Generating multiobjective trade-off: an algorithm for bicriteria problems, *Water Resource Research*, vol. 15, pp. 1001—1010, 1979.

[80] Cohon, J., *Multiobjective Programming and Planning*, Academic Press, New York, 1978.

[81] Cohoon, J., S. Hegde, and N. Martin, Distributed genetic algorithms for the floorplan design problem, *IEEE Transactions on Computer-Aided Design*, vol. 10, pp. 483—491, 1991.

[82] Coit, D. and A. Smith, Penalty guided genetic search for reliability design optimization, *International Journal of Computers and Industrial Engineering*, vol. 30, no. 4, pp. 895—904, 1996.

[83] Coit, D. and A. Smith, Reliability optimization of series-parallel systems using a genetic algorithm, *IEEE Transactions on Reliability*, 1996 (to appear).

[84] Conway, R., W. Maxwell, and L. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, MA, 1967.

[85] Cooper, L., Location-allocation problems, *Operations Research*, vol. 11, no. 3, pp. 331—344, 1963.

[86] Croce, F., R. Tadei, and G. Volta, A genetic algorithm for the job shop problem, *Computers and Operations Research*, vol. 22, pp. 15—24. 1995.

[87] Crowder, H. and M. Padberg, Solving large-scale symmetic traveling salesman problems to optimality, *Management Science*, vol. 26, pp. 495—509, 1980.

[88] Current, J. and M. Marsh, Multiobjective design of transportation networks and routing problems: taxonomy and annotation, *European Journal of Operational Research*, vol. 65, pp. 4—19, 1993.

[89] Current, J., and H. Min, Multiobjective design of transportation networks: taxonomy and annotation, *European Journal of Operational Research*, vol. 26, pp. 187—201, 1986.

[90] Dagtzig, G., D. Fulkerson, and S. Johnson, Solution of a large scale traveling salesman problems, *Operations Research*, vol. 2, pp. 393—410, 1954.

[91] Dannenbring, D., An evaluation of flow shop sequencing heuristics, *Management Science*, vol. 23, pp. 1174—1182, 1977.

[92] Dauzere-Pers, S. and J. Lasserre, A modified shifting bottleneck procedure for job-shop scheduling, *International Journal of Production Researches*, vol. 31, pp. 923—932, 1993.

[93] Davidor, Y., A genetic algorithm Applied to robot trajectory generation, in Davis [101], pp. 923—932, 1991.

[94] Davidor, Y., *Genetic Algorithms and Robotics*, World Scientific Publishing, Singapore, 1991.

[95] Davidor, Y., H. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature: PPSN III*, Springer-Verlag, Berlin, 1994.

[96] Davis, E, Computional experience with a new multi-resource algorithm, in Lombaers. H., editor, *Project Planning by Network Analysis*, pp. 256—260, North-Holland, New York, 1969.

[97] Davis, E. and G. Heidorn, An algorithm for optimal project scheduling under multiple resources constraints, *Management Science*, vol. 17, pp. B803—B816, 1971.

[98] Davis, E. and J. Patterson, A comparison of heuristic and optimum solutions in resource-constained project scheduling, *Management Science*, vol. 21, pp. 944—955, 1975.

[99] Davis, L., Applying adaptive algorithms to domains, *In Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 162—164, 1985.

[100] Davis, L., Job shop scheduling with genetic algorithms, in Grefenstette [186], pp. 136—140.

[101] Davis, L., editor, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[102] Davis, L., editor, *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, Los Altos, CA, 1987.

[103] Dawkins, R., *The Selfish Gene*, Oxford University Press, Oxford, 1976.

[104] De Jong, K., An Analsis of the Behavoir of a Class of Genetic Adaptive Systems, Ph. D. thesis, University of Michigan, Ann Arbor, 1975.

[105] De Jong, K., Genetic algorithms: a 25 year perspective, in Zurada et al. [433], pp. 125—134.

[106] De Jong, K. and W. Spears, On the state of evolutionary computation, in Forrest [137], pp. 618—623, 1993.

[107] Dell'Amico, M. and M. Trubian, Applying tabu search to the job shop scheduling problem, *Annals of Operations Research*, vol. 40, pp. 231—252, 1993.

[108] deSilva, C., editor, *Proceedings of the Second IEEE Conference on Evolutionary Computation*, IEEE Press, Perth, 1995.

[109] Dijkstra, E., A note on two problems in connection with graphs, *Numerische Mathematik*, vol. 1, pp. 269—271, 1959.

[110] Dileepan, P. and T. Sen, Bicriterion static scheduling research for a single machine, *Omega*, vol. 16, pp. 53—59, 1988.

[111] Domschke, W. and A. Drex, *An International Bibliography on Location and Layout Planning*, Springer, Heidelberg, 1984.

[112] Dorndorf, U. and E. Pesch, Evolution based learning in a job shop scheduling environment, *Computers and Operations Research*, vol. 22, pp. 25—40, 1995.

[113] Drexl, A. and J. Gruenewald, Nonpreemptive multi-mode resource constrained project scheduling, *IIE Transaction*, vol. 25, pp. 74—81, 1993.

[114] Dubois, D. and H. Prade, Fuzzy real algebra: some results, *Fuzzy Sets and Systems*, vol. 2, 327—348, 1979.

[115] Dudek, R., S. Panwalkar, and M. Smith, The lessons of flow shop scheduling research, *Operations Research*, vol. 40, pp. 7—13, 1992.

[116] Eilon, S. and I. Chowdhury, Minimizing waiting time variance in the single machine problem, *Management Science*, vol. 23, pp. 567—575, 1977.

[117] Eilon, S. et al., *Distribution Management: Mathematical Modeling and Practice Analysis*, Hafner, New York, 1971.

[118] El-Sayed, M. E. M., B. J. Ridgely, and Sandgren, Nonlinear structural optimization using goal programming, *Computers and Structures*, vol. 32, no. 1, pp. 69—73, 1989.

[119] Emmons, H., Scheduling to a common due date on parallel common processors, *Naval Research Logistics Quarterly*, vol. 34, pp. 803—810, 1987.

[120] Eshelman, L. J., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Francisco, 1995.

[121] Eshelman, L. and J. Schaffer, Real-coded genetic algorithms and intervalschemata, in Whitley [416], pp. 187—202.

[122] Falkenauer, E. and S. Bouffoix, A genetic algorithm for job shop, in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 824—829, 1991.

[123] Falkenauer, E. and Delchambre, A genetic algorithm for bin packing and line balancing, in *Proceeding of the IEEE International Conference on Robotics and Automation*, pp. 1186—1193, 1992.

[124] Fang, H., P. Ross, and D. Corne, A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems, in Forrest [137], pp. 375—382.

[125] Federgruen, A. and Z. Schechner, Cost formulas for continuous review inventory models with fixed delivery lags, *Operations Research*, vol. 31, no. 5, pp. 957—965, 1983.

[126] Fisher, H. and G. Thompson, Probabilistic learning combinations of job-shop scheduling rules, in Muth and Thompson [307], Chapter 15, pp. 1225—1251, 1963.

[127] Floudas, C. and P. Paralos, *Recent Advances in Global Optimization*, Princeton University Press, Princeton, 1992.

[128] Fogarty, T., editor, *Evolutionary Computing*, Springer-Verlag, Berlin, 1994.

[129] Fogel, D., editor, *Proceedings of the Third IEEE Conference on Evolutionary Computation*, IEEE Press, Nagoya, Japan, 1996.

[130] Fogel, D., An Introduction to simulated evolutionary optimization, *IEEE Transactions on Neural Networks*, vol. 5, pp. 3—14, 1994.

[131] Fogel, D., *Evolutionary computation: toward a new philosophy of machine intelligence*, IEEE Press, Piscataway, NJ, 1995.

[132] Fogel, D., editor, *Proceedings of the First IEEE Conference on Evolutionary Computation*, IEEE Press, Orlando, FL, 1994.

[133] Fogel, D. and W. Atmar, editors, *Proceeding of the First Annual Conference on Evolutionary Programming*, Evolutionary Programming Society, San Diego, 1992.

[134] Fogel, D. and W. Atmar, editors, *Proceeding of the Second Annual Conference on Evolutionary Programming*, Evolutionary Programming Society, La Jolla, 1993.

[135] Fonseca, C. and P. Fleming, Genetic algorithms for multiobjective optimization: formulation, discussion and generalization, in Forrest [137], pp. 416—423.

[136] Forrest, S., Documentation for prisoners dilemma and norms programs that use the genetic algorithm, working paper, University of Michigan, Ann Arbor, 1985.

[137] Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[138] Francis, R., L. McGinnis, and J. White, *Facility Layout and Location: An Analytical Approach*, 2nd ed.,

Prentice Hall, Englewood Cliffs, NJ, 2nd ed., 1992.

[139] Fraser, A., Simulation of genetic systems by automatic digital computers: I. introduction, *Australian Journal of Biological Science*, vol. 10, pp. 484－491, 1957.

[140] Fraser, A., Simulation of genetic systems by automatic digital computers: II. effects of linkage on rates of advance under selection, *Australian Journal of Biological Science*, vol. 10, pp. 492—499, 1957.

[141] Fraser, A., Simulation of genetic systems by automatic digital computers: VI. epistasis, *Australian Journal of Biological Science*, vol. 13, pp. 150—162, 1960.

[142] Fraser, A., Simulation of genetic systems, *Journal of Theoretical Biology*, vol. 2, pp. 329—346, 1962.

[143] Freeman, J., *Simulating Neural Networks with Mathematics*, Addison-Wesley, New York, 1994.

[144] French, S., *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, John Wiley & Sons, New York, 1982.

[145] Fyffe, D., W. Hines, and N. Lee, System reliability allocation and a computational algorithm, *IEEE Transactions on Reliability*, vol. R-17, pp. 64－69, 1968.

[146] Gabriete, D. and K. Ragsdell, Large scale nonlinear programming using the generalized reduced gradient method, *ASME Journal of Mechanical Design*, vol. 102, pp. 566—573, 1980.

[147] Garey, M. and D. Johnson, Strong NP- completeness results: motivation, examples and implications, *Journal of ACM*, vol. 25, pp. 499—508, 1978.

[148] Garey, M., D. Johnson, and R. Sethi, The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research*, vol. 1, pp. 117—129, 1976.

[149] Gen, M., Reliability optimization by 0-1 programming for a system with several failure modes, *IEEE Transactions on Reliability*, vol. R-24, pp. 206—210, 1975.

[150] Gen, M. and R. Cheng, Optimal design of system reliability under uncertainty using interval programming and genetic algorithm, Technical report, ISE94-6, Ashikaga Institute of Technology, Ashikaga, Japan, 1994.

[151] Gen, M. and R. Cheng, Interval programming using genetic algorithms, in *Proceeding of First International Symposium on Soft Computing for Industry*, 1996.

[152] Gen, M. and R. Cheng, A survey of penalty techniques in genetic algorithms, in Fogel [129], pp. 804—809.

[153] Gen, M. and K. Ida, *Linear Programming and Goal Programming using BASIC*, Denki-Shoin Publisher, Tokyo, 1984 (in Japanese).

[154] Gen, M. and K. Ida, *Goal Programming Using Turbo C*, HBJ Publishers, Tokyo, 1993 (in Japanese).

[155] Gen, M., K. Ida, E. Kono, and Y. Li, Solving bicriteria solid transportation problem by genetic algorithm, in Gen and Kobayashi [160], pp. 572—575.

[156] Gen, M., K. Ida, and J. Lee, Optimal selection and allocation of a system availability using 0-1 linear programming with GUB structure, *Transactions of Institute of Electronics, Information and Communication Engineers*, vol J71-D, pp. 2140—2147, 1988 (in Japanese).

[157] Gen, M., K. Ida, and Y. Li, Solving Bicriteria Solid Transportation Problem with Fuzzy Numbers by Genetic Algorithm, *International Journal of Computers and Industrial Engineering*, vol. 29, pp. 537—543, 1995.

[158] Gen, M., K. Ida, M. Sasaki, and J. Lee, Algorithm for solving large-scale 0-1 goal programming and its application to reliability optimization problem, *International Journal of Computers and Industrial Engineering*, vol. 17, pp. 525—530, 1989.

[159] Gen, M., K. Ida, and T. Taguchi, Reliability optimization problems: a novel genetic algorithm approach, Technical report, ISE93-5, Ashikaga Institute of Technology, Ashikaga, Japan, 1993.

[160] Gen, M. and T. Kobayashi, *Proceedings of the 16th International Conference on Computers and Industrial Engineering*, Ashikaga, Japan, 1994.

[161] Gen, M. and B. Liu, Evolution program for production plan problem, *Engineering Design and Automation*, vol. 1, no. 3, pp. 199—204, 1995.

[162] Gen, M. and B. Liu, A genetic algorithm for nonlinear goal programming, Technical report, ISE95-5, Ashikaga Institute of Technology, Ashikaga, Japan, 1995.

[163] Gen, M., B. Liu, and K. Ida, Evolution program for deterministic and stochastic optimizations, *European Journal of Operational Research*, 1996 (in press).

[164] Gen, M., B. Liu, K. Ida, and D. Zheng, Evolution program for constrained nonlinear optimization, in Gen and Kobayashi [160], pp. 576—579.

[165] Gen, M., Y. Tsujimura, and E. Kubota, Solving job-shop scheduling problem using genetic algorithms, in Gen and Kobayashi [160], pp. 576—579.

[166] Gen, M. and G. Zhou, An approach to the degree-constrained minimum spanning tree problem using genetic algorithm. Technical report ISE95-3, Ashikaga Institute of Technology, Ashikaga, Japan, 1995.

[167] Giffler, B. and G. Thompson, Algorithms for solving production scheduling problems, *Operations Research*, vol. 8, no. 4, pp. 487—503, 1960.

[168] Gilbert, E., Minimum cost communication networks, *Journal of Bell Systems Technology*, vol. 9, pp. 2209—2227, 1967.

[169] Gillies, A., Machine learning procedures for generating image domain feature detectors, Ph. D. thesis, University of Michigan, Ann Arbor, 1985.

[170] Glover, F. and H. Greenberg, New approaches for heuristic search: a bilateral linkage with artificial intelligence, *European Journal of Operational Research*, vol. 39, pp. 119—130, 1989.

[171] Goldberg, D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[172] Goldberg, D. and K. Deb, A comparative analysis of selection schemes used in genetic algorithms, in Rawlins [347], pp. 69—93, 1991.

[173] Goldberg, D., B. Korb, and K. Deb, Messy genetic algorithms: motivation, analysis, and first results, *Complex Systems*, vol. 3, pp. 493—530, 1989.

[174] Goldberg, D. and R. Lingle, Alleles, loci and the traveling salesman problem, in Grefenstette [186], pp. 154—159.

[175] Gong, D., M. Gen, W. Xu, and G. Yamazaki, Evolutionary strategy for obstacle location-allocation problem, in Zimmermann, H., editor, *Proceedings of the Third European Congress on Intelligent Techniques and Soft Computing*, pp. 426—433, Aachen, Germany, 1995.

[176] Gong, D., M. Gen, W. Xu, and G. Yamazaki, Hybrid evolutionary method for obstacle location-allocation problem, *International Journal of Computers and Industrial Engineering*, vol. 29, no. 1—4, pp. 525—530, 1995.

[177] Gong, D., M. Gen, G. Yamazaki, and W. Xu, Hybrid evolutionary method for capacitated location-allocation, *Engineering Design and Automation*, 1997 (to appear).

[178] Gong D., W. Xu, and M. Gen, Obstacle location-allocation in oil field, *Proceedings of 2nd Symposium of CIMS of Asian Countries*, Tokyo, 1994.

[179] Gordon, V. and D. Whitney, Serial and parallel genetic algorithms as functions optimizers, in Forrest [137], pp. 177—183.

[180] Gorges-Schleuter, M., ASPARAGOS, an asynchronous parallel genetic optimization strategy, in Schaffer [370], pp. 422—427.

[181] Graham, R. and P. Hell, On the history of the minimum spanning tree problem, *Annals of the History of Computing*, vol. 7, pp. 43—57, 1985.

[182] Graham, R., E. Lawler, J. Lenstra, and A. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling theory: a survey, *Annals of Discrete Mathematics*, vol. 5, pp. 287—326, 1979.

[183] Graves, S., A review of production scheduling, *Operations Research*, vol. 29, pp. 646—675, 1981.

[184] Greenberg, H., A branch-and-bound solution to the general scheduling problem, *Operations Research*, vol. 16, pp. 353—361, 1968.

[185] Grefenstette, J., R. Gopal, B. Rosmaita, and D. Gucht, Genetic algorithms for the traveling salesman problem, in Grefenstette [186], pp. 160—168.

[186] Grefenstette, J., editor, *Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

[187] Grefenstette, J., Optimization of control parameters for genetic algorithms, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, pp. 122—128, 1986.

[188] Grefenstette, J., Incorporating problem specific knowledge into genetic algorithms, in Davis [102].

[189] Grefenstette, J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[190] Grefenstette, J., Lamarkian learning in multi-agent environment, in Belew and Booker [30], pp. 303—310.

[191] Grefenstette, J., *Genetic Algorithms for Machine Learning*, Kluwer Academic Publishers, Norwell, MA, 1994.

[192] Grefenstette, J. and J. Baker, How genetic algorithms work: a critical look at implicit parallelism, in Schaffer [370], pp. 20—27.

[193] Grötschel, M., On the symmetric traveling salesman problem: solution of a 120-city problem, *Mathematical Programming Studies*, vol. 12, pp. 61—77, 1980.

[194] Grötschel, M. and O. Holland, Solution of large-scale symmetric traveling salesman problems, *Mathematical Programming*, vol. 51, pp. 141—202, 1991.

[195] Gupta, J., A functional heuristic algorithm for the flow shop scheduling problem, *Operations Research Quarterly*, vol. 22, pp. 39—47, 1971.

[196] Gupta, M., Y. Gupta, and A. Kumar, Minimizing flow time variance in a single machine system using genetic algorithm, *European Journal of Operational Research*, vol. 70, pp. 289—303, 1993.

[197] Gupta, S. and J. Kyparisis, Single machine scheduling research, *Omega*, vol. 15, pp. 207—227, 1987.

[198] Hadley, G., *Linear Programming*, Addison-Werley, Palo Alto, CA, 1962.

[199] Hakimi, S., Optimum distribution of switching centers in a communication network and some related graph theoretic problems, *Operations Research*, vol. 13, pp. 462—475, 1964.

[200] Hall, N., Single and multi-processor models for minimizing completion time variance, *Naval Research Logistics Quarterly*, vol. 33, pp. 49—54, 1986.

[201] Hall, N., W. Kubiak, and S. Sethi, Earliness-tardiness scheduling problems Ⅱ: deviation of completion times about a restrictive common due date, *Operations Research*, vol. 39, pp. 847—856, 1991.

[202] Hall, N. and M. Posner, Earliness-tardiness scheduling problems I: weighted deviation of completion times about a common due date, *Operations Research*, vol, 39, pp. 836—846, 1991.

[203] Halton, J., A retrospective and prospective survey of the Monte Carlo method *SIAM Review*, vol. 12, pp. 1—63, 1970.

[204] Hamacher, H. and S. Nickel, Restricted planar location problems and applications, *Naval Research Logistics*, vol. 42, pp. 967—992, 1995.

[205] Hammersley, J. and D. Handscomb, *Monte Carlo Methods*, Methuen, London, 1964.

[206] Hancock, P., An Empirical comparison of selection methods in evolutionary algorithms, in Fogarty [128], pp. 80—95.

[207] Hansen, E., *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, 1992.

[208] Hardy, G., J. Littlewood, and G. Polya, *Inequalities*, Cambridge University Press, London, 1967.

[209] Harris, B., A. Farhi, and J. Dutour, Aspects of a problem in clustering, Technical report, University of Pennsylvania, Philadelphia, 1972.

[210] Haupt, R., A survey of priority-rule based scheduling problem, *OR Spektrum*, vol. 11, pp. 3—16, 1989.

[211] Heragu, S. and A. Kusiak, Machine layout problem in flexible manufacturing systems, *Operations Research*, vol. 36, pp. 258—268, 1988.

[212] Hesser, J., R. Männer, and O. Stucky, Optimization of Steiner trees using genetic algorithms, in Schaffer [370], pp. 231—236.

[213] Hillier, F. and M. Connors, Quadratic assignment problem algorithms and the location of indivisible facilities, *Management Science*, vol. 13, pp. 42—57, 1966.

[214] Hillier, F. and G. Lieberman, *Introduction to Mathematical Programming*, McGraw-Hill, New York, 1991.

[215] Himmelblau, M., *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.

[216] Hinterding, R., Mapping, order-independent genes and the knapsack problem, in Fogel [132], pp. 13—17.

[217] Hitchcock, F., The distribution of a product from several sources to numerous locations, *Journal of Mathematical Physics*, vol. 20, pp. 224—230, 1941.

[218] Ho, J. and Y. Chang, A new heuristic for the n-job m-machine flow shop problem, *European Journal of Operational Research*, vol. 52, pp. 194—202, 1991.

[219] Hock, W. and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Springer-Verlag, New York, 1981.

[220] Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[221] Holsapple, C., V. Jacob, R. Pakath, and J. Zaveri, A genetics-based hybrid scheduler for generating static

schedules in flexible manufacturing contexts, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, pp. 953—971, 1993.

[222] Homaifar, A., C. Qi, and S. Lai, Constrained optimization via genetic algorithms, *Simulation*, vol. 62, no. 4, pp. 242—254, 1994.

[223] Horn, J., N. Nafpliotis, and D. Goldberg, A niched Pareto genetic algorithm for multiobjective optimization, in Fogel [132], pp. 82—87.

[224] Hwang, C. and A. Masud, *Multiple Objective Decision Making Methods and Applications*, Springer, Berlin, 1979.

[225] Hwang, C. and K. Yoon, *Multiple Attribute Decision Making: Methods and Applications*, Springer-Verlag, Berlin, 1981.

[226] Ida, K., M. Gen, and T. Yokota, System reliability optimization with several failure modes by genetic algorithm, in Gen and Kobayashi [160], pp. 349—352.

[227] Ignall, E. and L. Schrage, Application of the branch and bound technique to some flow shop scheduling problems, *Operations Research*, vol. 13, pp. 400—412, 1965.

[228] Ignizio, J. P., *Goal Programming and Extensions*, Heath, Lexington, MA, 1976.

[229] Ignizio, J., *Linear Programming in Single & Multiple-Objective System*, Prentice Hall, Englewood Cliffs, NJ, 1982.

[230] Ijiri, Y., *Management Goals and Accounting for Control*, North-Holland, Amsterdam, 1965.

[231] Ishbuchi, H. and H. Tanaka, Formulation and analysis of linear programming problem with interval coefficients, *Journal of Japan Industrial Management Association*, vol. 40, no. 5, 320—329, 1989 (in Japanese).

[232] Ishibuchi, H. and H. Tanaka, Multiobjective programming in optimization of the interval objective function, *European Journal of Operational Research*, vol. 48, pp. 219—225, 1990.

[233] Ishibuchi, H., N. Yamamoto, T. Murata, and H. Tanaka, Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems, *Fuzzy Sets and Systems*, vol. 67, pp. 81—100, 1994.

[234] Janilow, C. and Z. Michalewicz, An experimental comparison of binary and floating point representations in genetic algorithms, in Belew and Booker [30], pp. 31—36.

[235] Jeffcoat, D. E. and R. Bulfin, Simulated annealing for resource constrained scheduling, *European Journal of Operational Research*, vol. 70, pp. 43—51, 1993.

[236] Johnson, S., Optimal two-and-three stage production schedules with setup times included, *Naval Research Logistics Quarterly*, vol. 1, pp. 61—68, 1954.

[237] Joines, J. and C. Houck, On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs, in Fogel [132], pp. 579—584.

[238] Kaku, B. and G. Thompson, An exact algorithm for the general quadratic assignment problem, *European Journal of Operational Research*, vol. 23, pp. 383—390, 1986.

[239] Kall, P. and S. W. Wallace, *Stochastic Programming*, John Wiley & Sons, Chichester, 1994.

[240] Kanet, J., Minimizing the average deviation of job completion times about a common due date, *Naval Research Logistics Quarterly*, vol. 28, pp. 643—651, 1981.

[241] Kanet, J. and V. Sridharan, PROGENITOR: a genetic algorithm for production scheduling, *Wirtschaftsinformatik*, vol. 10, pp. 332—336, 1991.

[242] Katz, I. and L. Cooper, Facility location in the presence of forbidden regions; I. formulation and the case of the Euclidean distance with one forbidden circle, *European Journal of Operational Research*, vol. 6, pp. 166—173, 1981.

[243] Kaufmann, A. and M. Gupta, *Fuzzy Mathematical Models in Engineering and Management Science*, North-Holland, Amsterdam, 1988.

[244] Kaufmann, A, and M. Gupta, *Fuzzy Mathemational Models in Engineering and Management Science*, 2nd ed., North-Holland, Amsterdam, 1991.

[245] Kelley, J., The critical path method: resource planning and scheduling, in Muth and Thompson [307], pp. 347—365, 1963.

[246] Kennedy, S., Five ways to a smarter genetic algorithm, *AI Expert*, pp. 35—38, 1993.

[247] Kershenbaum, A., *Telecommunications Network Design Algorithms*, McGraw-Hill, New York, 1993.

[248] Kim, J. H., and H. Myung, A two-phase evolutionary programming for general constrained optimization prob-

lem, in *Proc. of the Fifth Annual Conference on Evolutionary Programming*, San Diego, 1996.

[249] Kobayashi, S., I. Ono, and M. Yamamura, An efficient genetic algorithm for job shop scheduling programs, in Eshelman [120], pp. 506—511.

[250] Koopmans, T. and M. Beckman, Assignment problems and the location of economic activities, *Econometrica*, vol. 25, pp. 53—76, 1957.

[251] Koulamas, C., The total tardiness problem: review and extensions, *Operations Research*, vol. 42, no. 6, pp. 1025—1044, 1994.

[252] Kouvelis, P., A. Kurawarwala, and G. Gutierrez, Algorithms for robust single and multiple period layout models for manufacturing systems, *European Journal of Operational Research*, vol. 63, pp. 287—303, 1992.

[253] Koza, John R., *Genetic Programming*, MIT Press, Cambridge, MA, 1992.

[254] Koza, John R., *Genetic Programming II*, MIT Press, Cambridge, MA, 1994.

[255] Kruskal, J., Jr., On the shortest spanning subtree of graph and the traveling salesman problem, *Proc. ACM*, vol. 7, no. 1, pp. 48—50, 1956.

[256] Kubota, A., Study on Optimal Scheduling for Manufacturing System by Genetic Algorithms, Master's thesis, Ashikaga Institute of Technology, Ashikaga, Japan, 1995.

[257] Kuo, T. and S. Hwang, A genetic algorithm with disruptive selection, in Forrest [137], pp. 65—69.

[258] Kusiak, A., *Intelligent Manufacturing System*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[259] Kusiak, A. and S. Heragu, The facility layout problem, *European Journal of Operational Research*, vol. 29, pp. 229—251, 1987.

[260] Lapaugh, A., Algorithms for Integrated Circuit Layout: Analytic Approach, Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1980.

[261] Larson, R. and G. Sadiq, Facility location with the manhattan metric in the presence of barriers to travel, *Operations Research*, vol. 31, pp. 652—669, 1983.

[262] Law, A. M. and W. D. Kelton, *Simulation Modeling and Analysis*, 2nd. ed., McGraw-Hill, New York, 1991.

[263] Lawer, E., J. Lenstra, A. Rinnooy Kan, and D. Shmoys, editors, *The Traveling Salesman Problem*, John Wiley & Sons, Chichester, 1985.

[264] Lawton, G., *A Practical Guide to Algorithms in C++*, John Wiley & Sons, New York, 1996.

[265] Lee, C., S. Danusaputro, and C. Lim, Minimizing weighted number of tardy jobs and weighted earliness-tardiness penalties about a common due date, *Computers and Operations Research*, vol. 18, pp. 379—389, 1991.

[266] Lee, C. and S. Kim, Parallel genetic algorithms for the tardiness job scheduling problem with general penalty weights, *International Journal of Computers and Industrial Engineering*, vol. 28, pp. 231—243, 1995.

[267] Lee, E. and R. Li, Comparison of fuzzy numbers based on the probability measure of fuzzy events, *Operations Research*, vol. 15, pp. 887—896, 1988.

[268] Lee, S., *Goal Programming for Decision Analysis*, Auerbach, PA, 1972.

[269] Lee, S. and D. Olson, A gradient algorithm for chance constrained nonlinear goal programming, *European Journal of Operational Research*, vol. 22, pp. 359—369, 1985.

[270] Lenstra, J., A. Rinnooy Kan, and P. Brucker, Complexity of machine scheduling problems, *Annals of Discrete Mathematics*, vol. 1, pp. 343—362, 1977.

[271] Li, C. and T. Cheng, The parallel machine min-max weighted absolute lateness scheduling problem, *Naval Research Logistics*, vol. 41, pp. 33—46, 1993.

[272] Leipins, G. and M. Hilliard, Genetic algorithm: foundations and applications, *Annals of Operations Research*, vol. 21, pp. 31—58, 1989.

[273] Liepins, G., M. Hilliard, M. Pallmer, and M. Morrow, Greedy genetics, in Grefenstette [189], pp. 90—99.

[274] Liepins. G., M. Hilliard, J. Richardson, and M. Pallmer, Genetic algorithm application to set covering and traveling salesman problems, in Brown, editor, *OR/AI: The Integration of Problem Solving Strategies*, 1990.

[275] Liepins, G. and W. Potter, A genetic algorithm approach to multiple fault diagnosis, in Davis [101], pp. 237—250.

[276] Liou, T. and M. Wang, Ranking fuzzy numbers with integral value, *Fuzzy Sets and Systems*, vol. 50, pp. 247—255, 1992.

[277] Liu, L., (s, S) continuous review models for inventory with random life-times, *Operations Research Letters*,

vol. 12, no. 3, pp. 161—167, 1990.

[278] Lozano-Perez, T. and M. Wesley, An algorithm for planning collision-free paths amongst polyhedral obstacles, Technical report, IBM Thomas J. Watson Research Center, 1978.

[279] Luenberger, D., *Linear and Nonlinear Programming*, 2nd. ed., Addison-Wesley, Reading, MA, 1984.

[280] Luss, H., Operations research and capacity expansion problems: a survey, *Operations Research*, vol. 30, no. 5, pp. 904—947, 1982.

[281] Manly, B. *The Statistics of Natural Selection on Animal Populations*, Chapman & Hall, London, 1984.

[282] Martello, S. and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester, 1990.

[283] Maza, M. and B. Tidor, An analysis of selection procedures with particular attention paid to proportional and Boltzmann selection, in Forrest [137], pp. 124—130.

[284] McDonnell, J., R. Reynolds, and D. Fogel, editors, *Evolutionary Programming IV*, MIT Press, Cambridge, MA, 1995.

[285] Männer, R. and B. Manderick, editors, *Parallel Problem Solving from Nature: PPSN II*, Elsevier Science Publishers, North-Holland, 1992.

[286] Merten, A. and M. Muller, Variance minimization in single machine sequencing problem, *Management Science*, vol. 18, pp. 518—528, 1972.

[287] Michalewicz, Z., *Genetic Algorithm + Data Structure = Evolution Programs*, 2nd ed., Springer-Verlag, New York, 1994.

[288] Michalewicz, Z., Genetic algorithms, numerical optimization, and constraints, in Eshelman [120], pp. 151—158.

[289] Michalewicz, Z., A survey of constraint handling techniques in evolutionary computation methods, in McDonnell et al. [284], pp. 135—155.

[290] Michalewicz, Z. and N. Attia, Evolutionary optimization of constrained problems, in Sebald and Fogel [375], pp. 98—108.

[291] Michalewicz, Z., D. Dasgupta, R. G. Le Riche, and Schoenauer, Evolutionary algorithms for Industrial Engineering problems, *International Journal of Computers and Industrial Engineering*, vol. 30, no. 4, 1996.

[292] Michalewicz, Z., T. Logan, and S. Swaminathan, Evolutionary operations for continuous convex parameter spaces, in Sebald and Fogel [375], pp. 84—97.

[293] Michalewicz, Z., and M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, *Evolutionary Computation*, vol. 4, no. 1, 1996 (in press).

[294] Michalewicz, Z., G. A. Vignaux and M. Hobbs, A non-Standard Genetic Algorithm for the Nonlinear Transportation Problems, *ORSA Journal on Computing*, vol. 3, no. 4, pp. 307—316, 1991.

[295] Miller, J., W. Potter, R. Gandham, and C. Lapena, An evaluation of local improvement operators for genetic algorithms, *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 23, no. 5, pp. 1340—1351, 1993.

[296] Mitchell, M., *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.

[297] Moinzadeh, K. and S. Nahmias, A continuous review model for an inventory system with two supply modes, *Management Science*, vol. 34, pp. 761—773, 1988.

[298] Morgan, B., *Elements of Simulation*, Chapman & Hall, London, 1984.

[299] Morton, T. and D. Pentico, *Heuristic Scheduling Systems-With Applications to a Production Systems and Project Management*, John Wiley & Sons, New York, 1993.

[300] Moscato, P. and M. Norman, A memetic approach for the traveling salesman problem: implementation of a computational ecology for combinatorial optimization on message-passing systems, in *Proceedings of the International Conference on Parallel Computing and Transportation Applications*, Amsterdam, 1992.

[301] Mühlenbein, H., How genetic algorithms really work: part I. mutation and hillclimbing, in Männer and Manderick [285], pp. 15—26.

[302] Mühlenbein, H. and D. Schlierkamp-Voosen, Predictive Models for the breeder genetic algorithm I. continuous parameter optimization, *Evolutionary Computation*, vol. 1, pp. 25—49, 1993.

[303] Mühlenbein, H., M. Schomisch, and J. Born, The parallel genetic algorithm as function optimizer, in Belew and Booker [30], pp. 271—278.

[304] Murtagh, B. and S. Niwattisyawong, An efficient method for the multi-depot location-allocation problem,

Journal of Operations Research Society, vol. 33, pp. 629—634, 1982.

[305] Murty, K, *Linear and Combinatorial Programming*, John Wiley & Sons, New York, 1976.

[306] Muselli, M., and S. Ridella, Global optimization of functions with the genetic algorithm, *Complex Systems*, vol. 6, pp. 193—212, 1992.

[307] Muth, J. and G. Thompson, editors, *Industrial Scheduling*, Prentice Hall, Englewood Cliffs, NJ, 1963.

[308] Myung, H., and J. H. Kim, Hybrid evolutionary programming for heavily constrained problems, *Bio-Systems*, vol. 38, pp. 29—43, 1996.

[309] Myung, Y., C. Lee, and D. Tcha, On generalized minimum spanning tree problem, *Networks*, vol. 26, pp. 231—241, 1995.

[310] Nakagawa, Y. and S. Miyazaki, Surrogate constraints algorithm for reliability optimization problems with two constraints, *IEEE Transactions on Reliability*, vol. 30, no. 2, pp. 175—180, 1981.

[311] Nakahara, Y., M. Sasaki, and M. Gen, On the linear programming with interval coefficients, *International Journal of Computers and Engineering*, vol. 23, pp. 301—304, 1992.

[312] Nakahara, Y., M. Sasaki, K. Ida, and M. Gen, A method for solving 0-1 linear programming problem with interval coefficients, *Journal of Japan Industrial Management Association*, vol. 42, no. 5, pp. 345—351, 1991 (in Japanese).

[313] Nakano, R. and T. Yamada, Conventional genetic algorithms for job-shop problems, in Belew and Booker [30], pp. 477—479.

[314] Narula, S. and C. Ho, Degree-constrained minimum spanning tree, *Computers and Operations Research*, vol. 7, pp. 239—249, 1980.

[315] Nawaz, M., E. Enscore, and I. Ham, A heuristic algorithm for the m-machine n-job flow shop sequencing problem, *Omega*, vol. 11, pp. 11—95, 1983.

[316] Nemhanser, G. and L. Wolsey, *Integer and Combinatorial Optimization*, Wiley-Interscience, New York, 1989.

[317] Norman, B. and J. Bean, Random keys genetic algorithm for job-shop scheduling; unabridged version, Technical report, University of Michigan, Ann Arbor, 1995.

[318] Norman, B. and J. Bean, Random keys genetic algorithm for scheduling, Technical report, University of Michigan, Ann Arbor, 1995.

[319] Ogbu, F and D. Smith, Simulated annealing for the flow-shop problem, *Omega*, vol. 19, pp. 64—67, 1991.

[320] Oliver, I., D. Smith, and J. Holland, A study of permutation crossover operators on the traveling salesman problem, in Grefenstette [89], pp. 224—230.

[321] Olsen, A., Penalty functions and the knapsack problem, in Fogel [132], pp. 554—558.

[322] Orvosh, D. and L. Davis, Using a genetic algorithm to optimize problems with feasibility constraints, in Fogel [132], pp. 548—552.

[323] Ostresh, L. Jr, An efficient algorithm for solving the two center location-allocation problem, *Journal of Regional Science*, vol. 15, pp. 209—216, 1975.

[324] Osyczka, A. and S. Kundu, A new method to solve generalized multicriteria optimization problems using genetic algorithm, *Structural Optimization*, vol. 10, no. 2, pp. 94—99, 1995.

[325] Otto, K. and E. Antonsson, Modeling imprecision in product design, in *Proceedings of the Third IEEE International Conference on Fuzzy Systems*, pp. 346—356, Yokohama, Japan, 1994.

[326] Padberg, M. and G., Rinaldi, Optimization of a 532-city symmetric traveling salesman problem by branch and cut, *Operations Research Letters*, vol. 6, pp. 1—7, 1987.

[327] Padberg, M. and G. Rinaldi, A branch and cut algorithm for the resolution of large scale symmetric traveling salesman problems, *SIAM Review*, vol. 33, pp. 60—100, 1991.

[328] Palmer, C., An Approach to a Problem in Network Design Using Genetic Algorithms, Ph. D. thesis, Polytechnic University, 1994.

[329] Palmer, C. and A. Kershenbaum, An approach to a problem in network design using genetic algorithms, *Networks*, vol. 26, pp. 151—163, 1995.

[330] Palmer, D. Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum, *Operations Research Quarterly*, vol. 16, pp. 101—107, 1965.

[331] Panwalkar, S. and W. Iskander, A survey of scheduling rules, *Operations Research*, vol. 25, pp. 45—61,

1977.

[332] Paredis, J., Exploiting constraints as background knowledge for genetic algorithms: a case-study for scheduling, in Männer and Manderick [285], pp. 281—290.

[333] Patterson, J. and G. Roth, Scheduling a project under multiple resource constraints: a 0-1 programming approach, *AIIE Transactions*, vol. 8, pp. 449—455, 1976.

[334] Patterson, J., F. Talbot, R. Slowinski, and J. Weglaz, Computational experience with a backtracking algorithm for solving a general class of precedence and resource constrained scheduling problems, *European Journal of Operational Research*, vol. 49, pp. 68—79, 1990.

[335] Pearson, D., N. Steele, and R. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms*, Springer-Verlag, New York, 1995.

[336] Pedrycz, W., Fuzzy modeling: methodology, algorithms and practice, in Zurada et. al. [433], Section 2.3, 1994,

[337] Picone, C. and W. Wilhelm, Perturbation scheme to improve Hillier's solution to the facilities layout problem, *Management Science*, vol. 30, pp. 1238—1249, 1984.

[338] Piggott, P. and F. Suraweera, Encoding graphs for genetic algorithms: an investigation using the minimum spanning tree problem, in Yao [463], pp. 305—314.

[339] Potvin, J. and D. Dube, Improving a vehicle routing heuristic through genetic search, in Fogel [132], pp. 194—199.

[340] Prim, R., Shortest connection networks and some generalizations, *Journal of Bell Systems Technology*, vol. 36, pp. 1389—1401, 1957.

[341] Pritsker, A., L. Waters, and P. Wolfe, Multiproject scheduling with limited resources: a 0-1 approach, *Management Science*, vol. 16, pp. 93—108, 1969.

[342] Prüfer, H., Neuer beweis eines satzes über permutation, *Arch. Math. Phys.*, vol. 27, pp. 742—744, 1918.

[343] Radcliffe, N., Genetic Neural Networks on MIMD Computers, Ph. D. thesis, University of Edinburgh, UK, 1990.

[344] Radcliffe, N. and P. Surry, Formal memetic algorithms, in Fogarty [128], pp. 1—16, 1994.

[345] Rai, S. and D. Agrawal, editors, *Distributed Computing Network Reliability*, IEEE Computer Society Press, Los Alamitos, CA, 1990.

[346] Ramakumar, R., *Engineering Reliability: Fundamentals and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[347] Rawlins, G., editor, *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[348] Reeves, C., Diversity and diversification in algorithms: some connections with tabu search, in Albrecht et al. [5], pp. 344—351.

[349] Reeves, C., Genetic algorithms and neighborhood search, in Fogarty [128], pp. 115—130.

[350] Reeves, C., A genetic algorithm for flow shop sequencing, *Computers and Operations Research*, vol. 22, pp. 5—13, 1995.

[351] Reeves, C. and C. Wright, An experimental design perspective on genetic algorithms, in Whitley and Vose [417], pp. 7—22.

[352] Reinelt, G., *The Traveling Salesman: Computational Solutions for TSP Application*, Springer-Verlag, Berlin, 1991.

[353] Renders, J. and H. Bersini, Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways, in Fogel [132], pp. 312—317.

[354] Richardson, J., M. Palmer, G. Liepins, and M. Hilliard, Some guidelines for genetic algorithms with penalty functions, in Schaffer [370], pp. 191—197.

[355] Ringuest, J. and D. Rinks, Interactive solutions for the linear multiobjective transportation problem, *European Journal of Operational Research*, vol. 32, pp. 96—106, 1987.

[356] Rinnooy Kan, A., *Machine Scheduling Problem: Classification, Complexity Computation*, Martinus Nijhoff, The Hague, 1976.

[357] Rosenblatt, M., The dynamics of plant layout, *Management Science*, vol. 32, pp. 76—86, 1986.

[358] Rosenblatt, M. and H. Lee, A robustness approach to facilities design, *International Journal of Production*

Research, vol. 25, pp. 479—486, 1987.

[359] Rosing, K., An optimal method for solving (generalized) multi-Weber problem, *European Journal of Operational Research*, vol. 58, pp. 414—426, 1992.

[360] Roy, B. and B. Sussmann, Les problemes d'ordonnancement avec constaintes disjonctives, Technical Report 9, SEMA, Note D.S., Paris, 1964.

[361] Rubinstein, R., *Simulation and the Monte Carlo Method*, John Wiley & Sons, New York, 1981.

[362] Saber, H. M. and A. Ravindran, Nonlinear goal programming theory and practice; a survey, *Computers and Operations Research*, vol. 20, no. 3, pp. 275—291, 1993.

[363] Sakawa, M., K. Kato, and T. Mori, Flexible scheduling in a machining center through genetic algorithms, *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 931—940, 1996.

[364] Sakawa, M., K. Kato, and T. Shibano, Fuzzy programming for multiobjective 0-1 programming problems through revised genetic algorithms, *European Journal of Operational Research* (in press).

[365] Sampson, S. and E. Weiss, Local search techniques for the generalized resource constrained project scheduling problem, *Naval Research Logistics*, vol. 40, pp. 665—675, 1993.

[366] Sannomiya, N. and H. Iima, Application of genetic algorithms to scheduling problems in manufacturing process, in Fogel [129], pp. 523—528, 1996.

[367] Sasaki, M., T. Yokota, and M. Gen, A method for solving fuzzy optimal reliability design problem by genetic algorithms, *Japanese Journal of Fuzzy Theory and Systems*, vol. 7, no. 5, pp. 1062—1072, 1995.

[368] Savelsbergh, M. and T. Volgenant, Edge exchanges in the degree-constrained spanning tree problem, *Computers and Operations Research*, vol. 12, pp. 341—348, 1985.

[369] Schaffer, J., Multiple objective optimization with vector evaluated genetic algorithms, in Grefenstette [186], pp. 93—100.

[370] Schaffer, J., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[371] Schrage, L., Minimizing the time-in-system variance for a finite job set, *Management Science*, vol. 21, pp. 540—543, 1975.

[372] Schwefel, H., *Numerical Optimization of Computer Models*, John Wiley & Sons, Chichester, 1981.

[373] Schwefel, H., *Evolution and Optimum Seeking*, John Wiley & Sons, New York, 1994.

[374] Schwefel, H. and R. Männer, editors, *Parallel Problem Solving from Nature*, Springer, New York, 1990.

[375] Sebald, A. and L. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, World Scientific Publishing, River Edge, NJ, 1994.

[376] Sen, T. and S. Gupta, A state-of-art survey of static scheduling research involving due dates, *Omega*, vol. 12, pp. 63—76, 1984.

[377] Shaefer, C., The ARGOT strategy: adaptive representation genetic optimizer technique, in Grefenstette [189], pp. 50—55.

[378] Shi, G., H. Iima, and N. Sannomiya, A method for contructing genetic algorithm in job shop problems, *Proc. of 8th SICE Symposium Decentralized Autonomous System*, pp. 175—178, 1996.

[379] Shiode, I., T. Nishida, and Y. Namasuya, Stochastic spanning tree problem, *Discrete Applied Mathematics*, vol. 3, pp. 263—273, 1981.

[380] Shore, R. and J. Tomkins, Flexible facilities design, *IIE Transactions*, vol. 12, pp. 200—205, 1980.

[381] Skiena, S., *Implementing Discrete Mathematics Combinatorics and Graph Theory with Mathematica*, Addison-Wesley, Reading, MA, 1990.

[382] Smith, A. and D. Tate, Genetic optimization using a penalty function, in Forrest [137], pp. 499—505.

[383] Sollin, M., Le trace de canalisation, in Berge, C. and A. Ghouilla-Houri, editors, *Programming, Games, and Transportation Networks*, John Wiley & Sons, New York, 1965.

[384] Solomon, M., The vehicle routing and scheduling problems with time windows constraints, *Operations Research*, vol. 35, pp. 254—265, 1987.

[385] Spears, W. and K. De Jong, On the virtues of parameterized uniform crossover, in Belew and Booker [30], pp. 230—236.

[386] Sprecher, A., *Recourse-Constrained Project Scheduling: Exact Methods for the Multi-Model Case*, Springer-Verlag, Berlin, 1993.

[387] Steinberg, L., The backboard wiring problem: a placement algorithm, *SIAM Review*, vol. 3, pp. 37—50, 1961.

[388] Stinson, J., E. Davis, and B. Khumawala, Multiple resource constrained scheduling using branch and bound, *AIIE Transactions*, vol. 10, pp. 252—259, 1978.

[389] Storer, R., S. Wu, and R. Vaccari, New search spaces for sequencing problems with application to job shop scheduling, *Management Science*, vol. 38, no. 10, pp. 1495—1510, 1992.

[390] Sundararaghravan, P. and M. Ahmed, Minimizing the sum of absolute lateness in single-machine and multimachine scheduling, *Naval Research Logistics Quarterly*, vol. 31, pp. 325—333, 1984.

[391] Syswerda, G., Uniform crossover in genetic algorithms, in Schaffer [370], pp. 2—9.

[392] Syswerda, G., Scheduling optimization using genetic algorithms, in Davis [101], pp. 332—349.

[393] Taillard, E., Benchmarks for basic scheduling problems, *European Journal of Operational Research*, vol. 64, pp. 278—285, 1993.

[394] Talbot, F. and J. Patterson, An efficient integer programming algorithm with network cuts for solving resource-constraints scheduling problems, *Management Sciences*, vol. 24, pp. 1163—1174, 1978.

[395] Tam, K., Genetic algorithms, function optimization, facility layout design, *European Journal of Operational Research*, vol. 63, pp. 322—346, 1992.

[396] Tamaki, H., M. Mori, and M. Araki, Generation of a set of pareto-optimal solutions by genetic algorithms, *Transactions of the Society of Instrument and Control Engineers*, vol. 31, no. 8, pp. 1185—1192, 1995 (in Japanese).

[397] Tamaki, H. and Y. Nishikawa, A paralleled genetic algorithm based on a neighborhood model and its application to the jobshop scheduling, in Männer and Manderick [285], pp. 573—582.

[398] Tate, D. and A. Smith, A genetic approach to the quadratic assignment problem, *Computers, and Operations Research*, vol. 22, pp. 73—83, 1995.

[399] Tate, D. and A. Smith, Unequal-area facility layout by genetic search, *IIE Transactions*, vol. 27, pp. 465—472, 1995.

[400] Thangiah, S. R. Vinayagamoorthy, and A. Guggi, Vehicle routing with time deadlines using genetic and local algorithms, in Forrest [137], pp. 506—513.

[401] Thierens, D. and D. Goldberg, Convergence models of genetic algorithm selection schemes, in Davidor et al. [95], pp. 119—129.

[402] Tillman, F. C. Hwang, and W. Kuo, *Optimization of Systems Reliability*, Marcel Dekker, New York, 1980.

[403] Tillman, F. Optimization by integer programming of constrained reliability problems with several modes of failure, *IEEE Transactions on Reliability*, vol. R-18, pp. 47—53, 1969.

[404] Tsujimura, Y., M. Gen, and R. Cheng, An efficient method for solving traveling salesman problems with advanced genetic algorithms, *Transactions of the Institute of Electronics, Information and Computer Engineering*, submitted for publication, 1997 (to appear).

[405] Tsujimura, Y., M. Gen, and E. Kubota, Flow-shop scheduling with fuzzy processing time using genetic algorithms, *The 11th Fuzzs Systems Symposium*, pp. 248—252, Okinawa, 1995 (in Japanese).

[406] Tsujimura, Y., M. Gen, and E. Kubota, Solving job-shop scheduling problem with fuzzy processing time using genetic algorithm, *Japanese Journal of Theory and Systems*, vol. 7, pp. 1073—1083, 1995.

[407] Tanino, T., M. Tanaka, and C. Hojo, An interactive multicriteria decision making method by using a genetic algorithm, in *Proceedings of International Conference on System Science and System Engineering*, pp. 381—386, 1993.

[408] Van de Panne, C. and W. Popp, Minimum cost cattle feed under probabilistic protein constraints, *Management Science*, vol. 9, pp. 405—430, 1963.

[409] Van Laarhoven, P., E. Aarts, and J. Lenstra, Job shop scheduling by simulated annealing, *Operations Research*, vol. 40, no. 1, pp. 113—125, 1992.

[410] Vignaux, G. A. and Z. Michalewicz, A Genetic Algorithm for the Linear Transportation Problem, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, pp. 445—452, 1991.

[411] Wang, P., G. Wilson, and N. Odrey, An on-line controller for production system with seasonal demands, *International Journal of Computers and Industrial Engineering*, vol. 27, pp. 565—574, 1994.

[412] Weistroffer, H., An interactive goal programming method for nonlinear multiplecriteria decision-making prob-

lems, *Computers and Operations Research*, vol. 10, no. 4, pp. 311—320, 1983.

[413] Wetzel, A., Evaluation of the effectiveness of genetic algorithms in combinatorial optimization, Technical report, University of Pittsburgh, 1983.

[414] Whitley, D., GENITOR: a different genetic algorithm, in *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver, 1989.

[415] Whitley, D., V. Gordan, and K. Mathias, Lamarckian evolution, the Baldwin effect and function optimization, in Davidor et al. [95], pp. 6—15.

[416] Whitley, L., editor, *Foundations of Genetic Algorithms* 2, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[417] Whitley, L. and M. Vose, editors, *Foundations of Genetic Algorithms* 3, Morgan Kauffmann Publishers, San Mateo, CA, 1995.

[418] Winter, G., et al, *Genetic Algorithms in Engineering and Computer Science*, John Wiley & Sons, New York, 1996.

[419] Winston, W., *Operations Research: Appliciations and Algorithms*, 3rd ed., Duxbury Press, Belmont, CA, 1994.

[420] Wong, D. and C. Liu, A new algorithm for floorplan design, in Wizard, V and M. Yannakakis, editors, *Proceedings of the 23rd ACM-IEEE Design Automation Conference*, pp. 101—107, Boston, 1986.

[421] Wright, A., Genetic algorithms for real parameter optimization, in Rawlins [347], Chapter 4.

[422] Xu, W., On the quadratic minimum spanning tree problem, in Gen, M. and W. Xu, editors, *Proceedings of 1995 Japan-China International Workshops on Information Systems*, pp. 141—148, Ashikaga, Japan, 1995.

[423] Yamada, T. and R. Nakano, A genetic algorithm applicable to large-scale job-shop problems, in Mãnner and Manderick [285], pp. 281—290.

[424] Yamada, T. and R. Nakano, Genetic algorithm and job-shop scheduling problem, *Systems, Control and Information*, vol. 37, no. 8, pp. 484—489, 1993 (in Japanese).

[425] Yamamura, M., T Ono, and S. Kobayashi, Character-preserving genetic algorithms for traveling salesman problem, *Journal of Japan Society for Artificial Intelligence*, vol. 6, pp. 1049—1059, 1992.

[426] Yang, X. and M. Gen, Evolution program for bicriteria transportation problem, in Gen and Kobayashi [160], pp. 451—454.

[427] Yokota, T., M. Gen, and K. Ida, System reliability of optimization problems with several failure modes by genetic algorithm, *Japanese Journal of Fuzzy Theory and Systems*, vol. 7, no. 1, pp. 117—185, 1995.

[428] Yokota, T., M. Gen, K. Ida, and T. Taguchi, Optimal design of system relibility by an approved genetic algorithm, *Transactions of Institute of Electronics, Information and Communication Engineers*, vol. J78A, no. 6, pp. 702—709, 1995.

[429] Zadeh, L., Fuzzy sets as a basis for a theory of possibility, *Fuzzy Sets and Systems*, vol. 1, pp. 3—28, 1978.

[430] Zhang, L. and W. Zheng, Remodeling the film-copy deliverer problem, in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pp. 543—547, San Antonio, 1994.

[431] Zhang, L. and W. Zheng, The extended traveling salesman problem and film copy deliver problem, *The Sixth Annual International Symposium on Algorithms and Computation*, Callaghan, Australia, 1995.

[432] Zimmerman, H., *Fuzzy Set Theory and Its Applications*, 2nd ed., Academic Publishers, Norwell, MA, 1991.

[433] Zurada, J., R. Marks II, and C. Robinson, editors, *Computational Intelligence: Imitating Life*, IEEE Press, New York, 1994.

[434] Alliot, J. M., E. Lutton, E. Ronald, M. Schoenauer, and D. Snyders, editors, *Artificial Evolution: European Conference, AE'95, Brest*, Springer-Verlag, Berlin, 1996.

[435] Fogel, L., P. J. Angeline, and T Bäck, editors, *Proceedings of the 5th Annual Conference on Evolutionary Programming*, MIT Press, Cambridge, MA, 1996.

[436] Angeline, P. and K. E. Kinnerar, Jr., editors, *Advances in Genetic Programming*, vol. 2, MIT Press, Cambridge, MA, 1996.

[437] Bäck, T. and H. Schwefel, Evolutionary computation: an overview, in Fogel [129], pp. 20—29.

[438] Ebeling, W. and H.-M. Voigt, editors, *Proceedings of the 4th Conference on Parallel Problem Solving from Nature*, Springer-Verlag, Berlin, 1996.

[439] Fogel, D. and A. Ghozeil, Using fitness distributions to design more efficient evolutionary computations, in

Fogel [129], pp. 11—19.

[440] Furuhashi, T., editor, *Advances in Fuzzy Logic, Neural Networks and Genetic Algorithms*, Springer-Verlag, Berlin, 1995.

[441] Gen, M. and K. Ida, editors, *Proceedings of Mini-Symposium on Genetic Algorithms and Engineering Design*, Ashikaga, Japan, 1996.

[442] Gen, M. and Y. Tsujimura, editors, *Evolutionary Computations and Intelligent Systems*, Gordon & Breach Publishers, NJ, 1997.

[443] Gen, M. and R. Cheng, A hybrid search for machine scheduling problems, in Zimmermann [469], pp. 378—383.

[444] Gen, M. and B. Liu, Evolution program for optimal capacity expansion, *Journal of Operations Research of Japan*, 1997, in press.

[445] Gong, D., G. Yamazaki, and M. Gen, Evolutionary program for optimal design of material distribution system, in Fogel [129], pp. 139—143.

[446] Gong, D., M. Gen, and G. Yamazaki, Evolutionary method for facility location problem, in Zimmermann [469], pp. 373—377.

[447] Grierson, D. and P. Hajela, editors, *Emergent Computing Methods in Engineering Design: Applications of Genetic Algorithms and Neural Networks*, Springer-Verlag, Berlin, 1996.

[448] Herrera, H. and J. L. Verdegay, editors, *Genetic Algorithms and Soft Computing*, Physica-Verlag, Heidelberg, 1996.

[449] Koza, J. R., editor, *Genetic Programming: Proceedings of the First Annual Conference*, MIT Press, Cambridge, MA, 1996.

[450] Langton, C. and T. Shimohara, editors, *Artificial Life V: the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, Nara, 1996.

[451] Li, Y., M. Gen, and K. Ida, Evolutionary computation for multicriteria solid transportation problem with fuzzy numbers, in Fogel [129], pp. 596—601.

[452] Michalewicz, Z. and M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, *Evolutionary Computation* (to appear).

[453] Michalewicz, Z., Evolutionary computation: practical issues, in Fogel [129], pp. 30—39.

[454] Myung, H. and J. Kim, Constrained optimization using two-phase evolutionary programming, in Fogel [129], pp. 262—267.

[455] Sakawa, M., *Fuzzy Sets and Interactive Multiobjective Optimization*, Plenum Press, New York, 1993.

[456] Sakawa, M. and M. Tanaka, *Genetic Algorithms*, Asakura Syoten, Tokyo, 1995 (in Japanese).

[457] Schniederjans, M., *Goal Programming*, Kluwer Academic, Dordrecht, Germany, 1995.

[458] Singh, N., *Systems Approach to Computer-Integrated Design and Manufacturing*, John Wiley & Sons, New York, 1996.

[459] Tamaki, H., H. Kita, and S. Kobayashi, Multi-objective optimization by genetic algorithms: a review, in Fogel [129], pp. 517—522.

[460] Tsujimura, Y and M. Gen, Genetic algorithms for solving multi-processor scheduling problems, in Yao, Kim, and Furuhashi [464].

[461] Yamamura, M., I. Ono, and S. Kobayashi, Emergent search on double circle TSPs using subtour exchange crossover, in Fogel [129], pp. 535—540.

[462] Yao, X., editor, *Progress in Evolutionary Computation*, Springer-Verlag, Berlin, 1995.

[463] Yao, X., editor, *Evolutionary Computation: Theory and Applications*, World Scientific Publishing, Singapore, 1996.

[464] Yao, X., J. H. Kim, and T. Furuhashi, editors, *Proceeding of the First Asia-Pacific, Conference on Simulated Evolution and Learning*, Taejon, 1996.

[465] Yokota, T., M. Gen, and Y. X. Li, Genetic algorithms for nonlinear mixed integer programming problems and its applications, *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 905—917, 1996.

[466] Zhao, L., Y. Tsujimura, and M. Gen, Genetic algorithms for fuzzy clustering, in Fogel [129], pp. 716—719.

[467] Zhou, G. and M. Gen, The genetic algorithms approach to the multicriteria minimum spanning tree problem, in Yao, Kim and Furuhashi [464].

[468] Zhou, G. and M. Gen, An effective genetic algorithm approach to the quadratic minimum spanning tree problem. in Yao, Kim and Furuhashi [464].

[469] Zimmerman, H., editor, *Proceedings of Fourth European Congress on Intelligent Techniques and Soft Computing*, Aachen, 1996.

[470] Hitomi, K. *Manufacturing Systems Engineering*, 2nd ed., Taylor & Francis, London, 1996.

[471] Higuchi, T., D. Mange, H. Kitano, and H. Iba, editors, *Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware*, Springer-Verlag, Berlin, 1996.